Ramaiah Institute of Technology

(Autonomous Institute, Affiliated to VTU)

Department of Electronics and Communication Engineering

VII Semester Microcontroller Lab (ECL7 )

**Contents**

**1. Introduction to** **Software IAR embedded workbench**.

**2. Programs involving Data Transfer Instructions**

**3. Programs involving Arithmetic Operation**

**4. Programs involving Bit Manipulation Instructions**.

**5. Programs to perform Mathematical Operations**

**6. Code Conversion**

**7. Programs Involving Branch /Loop Instruction**

**8. Programs Involving String Manipulation**

**9. Programs to interface modules to MSP430 microprocessor**

| **Lab Session No** | **Experiments** | **CO** |
| --- | --- | --- |
|  | **Introduction to** **Software IAR embedded workbench**.  with simple example | CO1,5 |
|  | ARM assembly language programs for data transfer, arithmetic, Thumb instructions and logical operations. | CO1,2,5 |
|  | Sort a given set of ‘N’, 16-bit numbers in ascending order using bubble sort algorithm | CO2,5 |
|  | Reverse a given string and verify whether it is a palindrome or not. Display the appropriate message. | CO2,5 |
|  | Compute nCr using recursive procedure. Assume that „n‟ and „r‟ are non-negative integers. | CO2, 5 |
|  | Search a key element “X” in a list of „n‟ 16-bit numbers using binary search algorithm | CO2, 5 |
|  | C Programs for matrix multiplication, matrix addition and sparse matrix implementation. | CO3, 5 |
|  | **Programs to interface modules to LPC2148 microcontroller**  Familiarize I/O ports of LPC 2148 – on/off control of LEDs using switches. | CO3,5 |
|  | Interface keypad and display the key pressed on LCD. | CO3, 4 |
|  | Waveform generation using the internal DAC of LPC 2148. |  |
|  | Convert a given analog voltage to digital using ADC of LPC 2148. |  |
|  | Design and display a 4 digit counter. |  |

**Experiment 1: Introduction to** **Software IAR embedded workbench**.

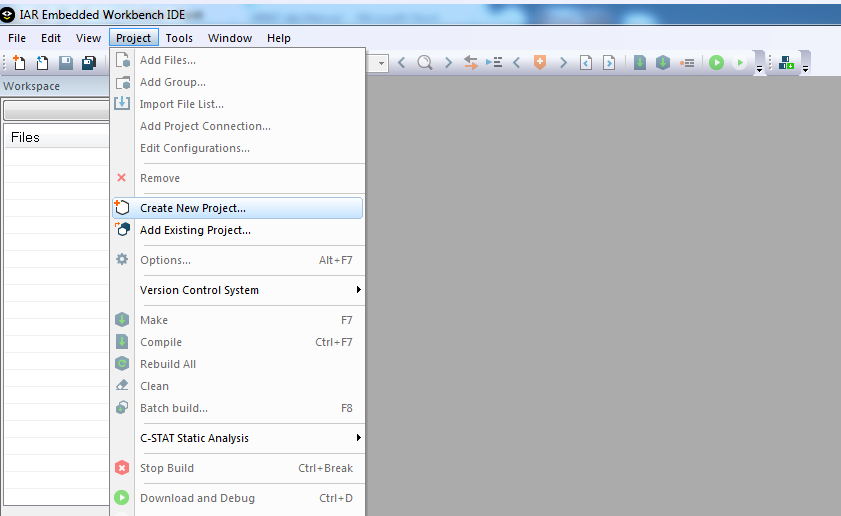
**CREATION OF NEW PROJECT and SIMULATING:**

The **IAR** compiler is used to create,compile & simulate the projects for ARM family micro-controllers. The procedure to create projects is as follows

Step 1:Create a folder with the name related to the project in any drive.

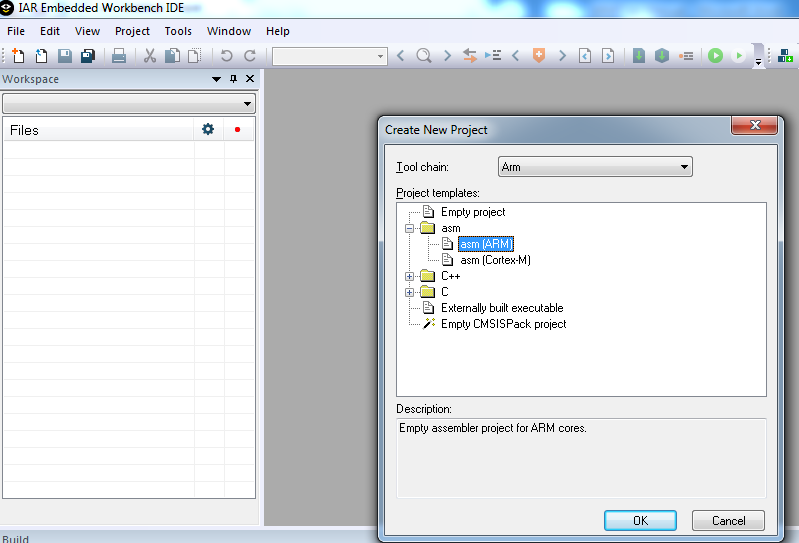
Step 2: Open **IAR Embedded Workbench** software. A new workspace launcher will open.

Step 3: Go to **Project** click “**Create** **New** **Project”.**

****

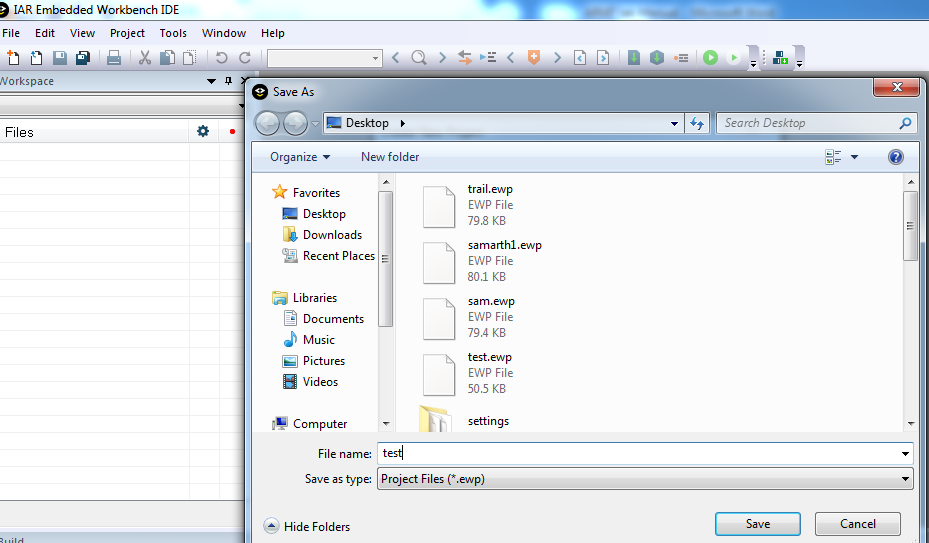
Step 4: Tool chain:Select **ARM**

Project templates : select **asm (ARM)**  if assembly code you would like write, else you would like to write **C** then select **main**

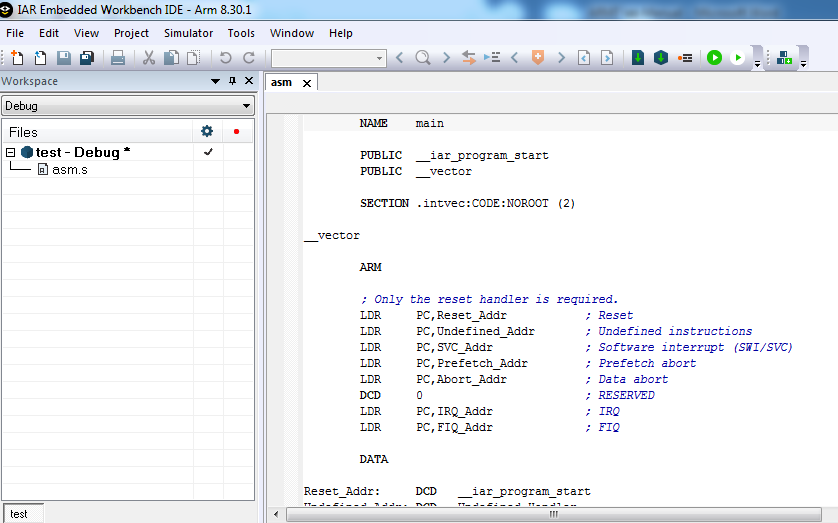


Step 5: select **asm** and click **OK.**

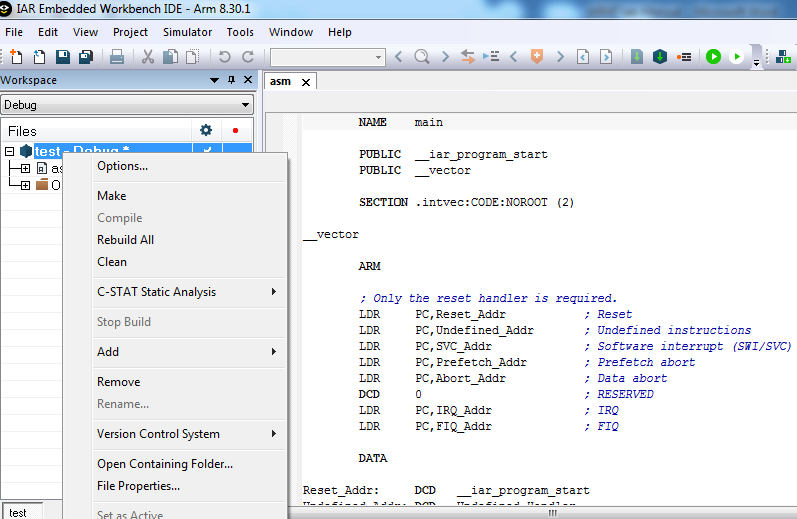
Step 6: Give project name and save the file name in corresponding folder.



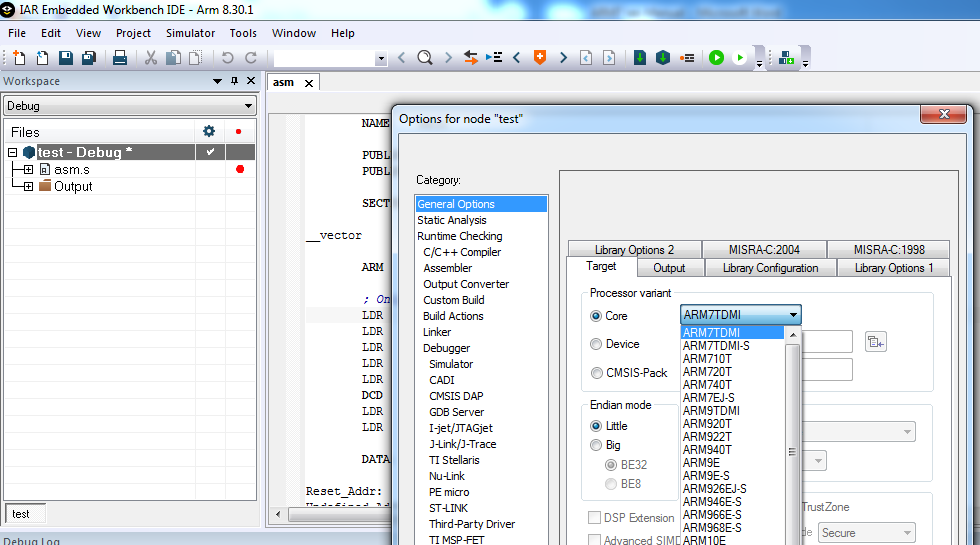
Step7: The window you can see after step 6**.**



Step 8: Go to project name, right click and select **options.**

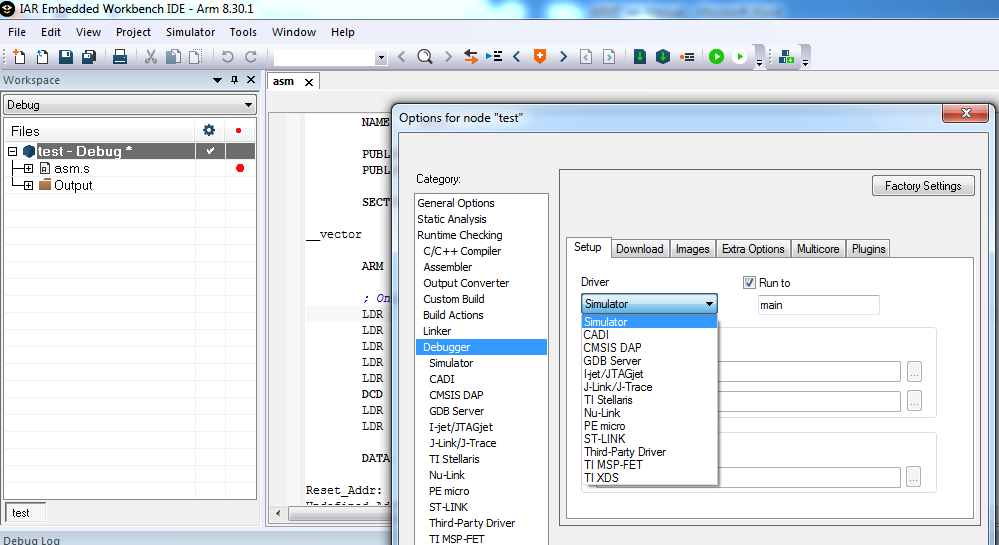


Step 9: Go to **general** **options** > **target** **tab**> **device**>select device **MSP430F4618**



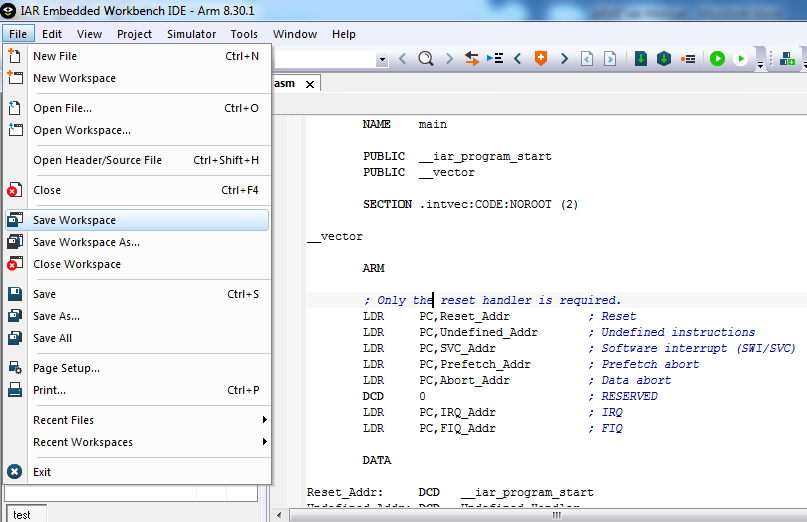
Step 10: Select device **ARM7TDMI** and click **OK.**

Step 11: Go to **debugger** > **setup** tab > select **simulator** > click **OK**

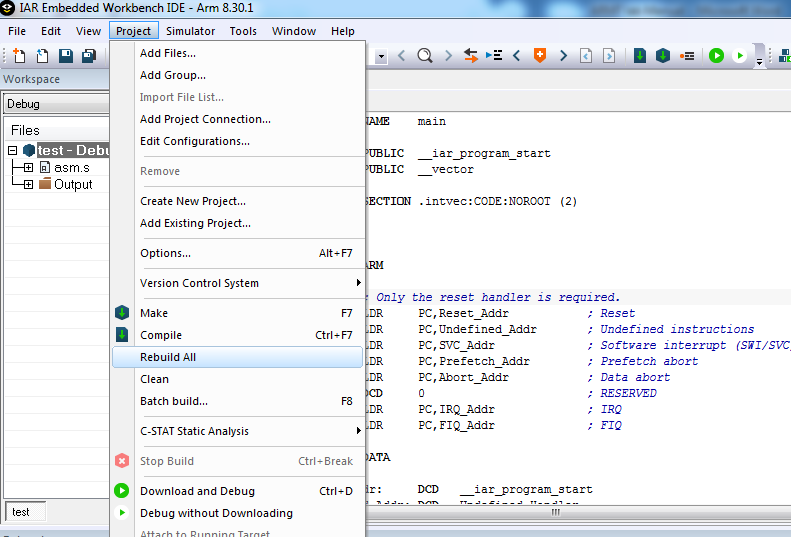


Step 12: Type program in editor window.

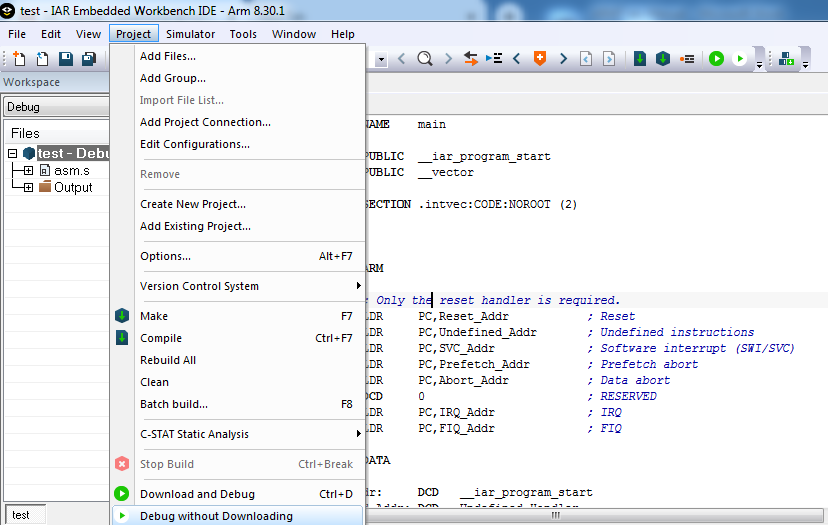
Step 13: Go to **file** -> **Save** **Workspace**, save the workspace in corresponding folder, Click **ok.**



Step 14: Go to project-> **Rebuild** **All,**if any **errors** or **warnings** you will get the result in **build** console

****

Step 15: To **simulate** the program: Go to **Project** 🡪 **Debug without Downloading**



Step 16: PC points to the initial point of the program.

Step 17:

1. If you want to see the memory then go to **view**->**memory.**
2. If you want to see the resistor then go to **view**->**register.**
3. Any resistor you can add to the watch window. Just right click on the register and click **add to watch.**
4. If you need to perform **Run**, **step over**, **step into**, **step out** and **next statement**

Then go to the **Debug** and click your option.

**Experiment 2: ARM assembly language programs for data transfer, arithmetic, Thumb instructions and logical operations.**

NAME main

PUBLIC \_\_iar\_program\_start

# SECTION .intvec:CODE:NOROOT (2)

# SECTION .data:CONST(2)

# NUM

# DC16 0X1111, 0X2222, 0X3333,0X4444,0X5555

# NUM2

# DS16 5

# \_\_iar\_program\_start

# main

# MOV R0,#05

# ADR R9, NUM

# ADR R7, NUM2

# LOOP LDRH R5,[R9]

# STRH R5,[R7]

# ADD R9,R9,#2

# ADD R7,R7,#2

# SUBS R0,R0,#1

# BNE LOOP

# stop B stop

# END

# Assignment: Exchange numbers between two memory locations

# Addition and Subtraction

NAME    main

        PUBLIC  \_\_iar\_program\_start

        SECTION .intvec:CODE:NOROOT (2)

\_\_iar\_program\_start

main

          LDR R0,=0XF631024C

          LDR R1,=0X27539ABD

          LDR R3,=0X45107CD2

          LDR R4,=0XF34567FC

          ADDS R5,R0,R3

          ADCS R6,R1,R4

          ADDCS R2,R2,#00000001

          SUBS R7,R0,R3

          SUBCS R8,R1,R4

          ADDCC R9,R9,#0XFFFFFFFF

STOP B STOP

        END

# Find the sum of squares of given N numbers

NAME    main

     PUBLIC  \_\_iar\_program\_start

        SECTION .data:CONST(2)

NUM

        DC16  0XFFFF, 0XFFFF, 0XFFFF

NUMEND

        DC32 0

LENGTH

        DC32    (NUMEND-NUM)/2

RESULT

        DS32 2

        PUBLIC  \_\_iar\_program\_start

        SECTION .intvec:CODE:NOROOT (2)

\_\_iar\_program\_start

main

    LDR R0,LENGTH

    ADR R1,NUM

    MOV R8,#0

    MOV R9,#0

LOOP    LDRH R4,[R1]

    UMULL R3,R7,R4,R4

    ADDS R8,R8,R3

    ADCCS  R9,R9,R7

    ADD R1,R1,#2

    SUBS R0,R0,#1

    BNE LOOP

    STR R9,RESULT

    STR R8,RESULT+1

STOP B STOP

    END

# Convect Packed BCD to hex

NAME main

        PUBLIC \_\_iar\_program\_start

        SECTION .data:CONST:NOROOT(2)

NUM DC16 0X1234

RESULT DS32 1

        SECTION .intvec:CODE:NOROOT(2)

\_\_iar\_program\_start

main

     ADR R0,NUM

     MOV R5,#4

     MOV R9,#12

     MOV R1,#0

     EOR R2,R2,R2

     MOV R10,#10

     MOV R11,#0X0F

LOOP MOV R7,R1

     MUL R1,R7,R10

     LDRH R8,[R0]

     ROR R8,R8,R9

     AND R4,R8,R11

     ADD R1,R1,R4

     SUBS R9,R9,#4

     SUBS R5,R5,#1

     BNE LOOP

     STR R1,RESULT

STOP B STOP

        END

# Convert Unpacked BCD to hex

NAME main

        PUBLIC \_\_iar\_program\_start

        SECTION .data:CONST:NOROOT(2)

string DCB 01,02,03,04

end DCB 0x00

length DCB end-string

copy DS8 8

        SECTION .intvec:CODE:NOROOT(2)

\_\_iar\_program\_start

main

     Ldr R5,length

     subs r5,r5,#1

     Ldr R0,=string

     ldr R4,=copy

     mov R1,R0

     add R1,R1,#1

 LDRB R3,[R0],#1

     MOV R7,#0XA

loop LDRB R6,[R1],#1

     MUL R3,R7,R3

     ADD R3,R3,R6

     STR R3,[R4]

     SUBS R5,R5,#1

     BNE loop

stop b stop

        end

# Find the largest element in a given array of N numbers

NAME    main

        PUBLIC  \_\_iar\_program\_start

        SECTION .data:CONST:NOROOT (2)

mydata:

        DC32 0X99999999,0X33333333,0X44444444,0XAAAAAAAA,0XFFFFFFFF

mydataend:

        DC32 0x00

length:

        DC32 (mydataend-mydata)/4

        SECTION .intvec:CODE:NOROOT (2)

\_\_iar\_program\_start

MAIN

        LDR R0,length

        SUB R0,R0,#1

        MOV R5,R0

        LDR R1,=mydata

LOOP    LDR R3,[R1]

        LDR R4,[R2]

        CMP R3,R4

        BHS LOOP2

blo loop1

        stop b stop

LOOP1   add r2,r2,#4

add r1,r1,#4

mov r10,r3

subs r0,r0,#1

bhi loop

stop b stop

LOOP2   add r2,r2,#4

add r1,r1,#4

mov r9,r3

subs r0,r0,#1

bhi loop

stop b stop

end

# Assignment: Find the smallest element in a given array of N numbers

# Find the factorial of a given number

NAME    main

  PUBLIC  \_\_iar\_program\_start

        SECTION .intvec:CODE:NOROOT (2)

\_\_iar\_program\_start

main

        mov r0,#10

        mov r1,r0

loop    subs r1,r1,#1

        mulne r2,r0,r1

        mov r0,r2

        bne loop

stop    B       stop

        END

**Copy and reverse a string**

NAME main

        PUBLIC \_\_iar\_program\_start

        SECTION .data:CONST:NOROOT(2)

string DCB 'FUN'

endDCb 0x00

length DCD end-string

rev DS8 8

        SECTION .intvec:CODE:NOROOT(2)

\_\_iar\_program\_start

main

     Ldr R5,length

     Ldr R0,=string

     ldr R4,=rev

     mov R1,R0

     add R1,R1,R5

     SUB R1,R1,#1

loop CMP R1,R0

     BMI stop

     LDRB R3,[R1],#-1 (ldrb  r3,[r0],#1)

     STRB R3,[R4],#1

     B loop

stop b stop

        end

**Experiment 3: Sort a given set of ‘N’, 16-bit numbers in ascending order using bubble sort algorithm**

NAME    main

        PUBLIC  \_\_iar\_program\_start

        SECTION .data:CONST:NOROOT (2)

mydata:

        DC32 0X99999999,0X33333333,0X44444444,0XAAAAAAAA,0XFFFFFFFF

mydataend:

        DC32 0x00

length:

        DC32 (mydataend-mydata)/4

        SECTION .intvec:CODE:NOROOT (2)

\_\_iar\_program\_start

MAIN

        LDR R0,length

        SUB R0,R0,#1

        MOV R5,R0

        LDR R1,=mydata

LOOP2   MOV R0,R5

        MOV R2,R1

        ADD R2,R2,#4

L1      LDR R3,[R1]

        LDR R4,[R2]

        CMP R3,R4

        BHS LOOP1 (blo)

        SWP R3,R4,[R1]

        STR R3,[R2]

LOOP1   ADD R2,R2,#4

        SUBS R0,R0,#1

        BHI L1

        ADD R1,R1,#4

        SUBS R5,R5,#1

        BGT LOOP2

stop    B    stop

        END

**Assignment: Sort a given set of ‘N’, 16-bit numbers in descending order using bubble sort algorithm**

**Experiment 4: Reverse a given string and verify whether it is a palindrome or not. Display the appropriate message.**

NAME main

        PUBLIC \_\_iar\_program\_start

        SECTION .data:CONST:NOROOT(2)

string DCB 'radar'

endDCb 0x00

length DCD end-string

        SECTION .intvec:CODE:NOROOT(2)

\_\_iar\_program\_start

main

     ldr R5,length

     ldr R0,=string

     mov R1,R0

     add R1,R1,R5

     SUB R1,R1,#1

loop CMP R1,R0

     BMI PALI

     LDRB R2,[R0],#1

     LDRB R3,[R1],#-1

     CMP R2,R3

     BEQ loop

NOTPALI MOV R10,#0

STOP B STOP

PALI MOV R10,#0XFF

stop    b stop

        end

**Experiment 5: Compute nCr using recursive procedure and number system conversions.**

NAME   main

        PUBLIC  \_\_iar\_program\_start

        SECTION .data:CONST:NOROOT (2)

n    DC32  0xa

r    dc32  0x3

quotient dcd 0x0,0x0,0x0

remain   dcd 0x0

SECTION .intvec:CODE:NOROOT (2)

 \_\_iar\_program\_start

main

                 ldr r0,n

                bl fact            ;N! IS FOUND

                mov r10,r6 ;r10=n!

                ldr r0,r

                bl fact           ; R! IS FOUND

                mov r1,r6 ;r1=r!

                ldr r5,n

                ldr r8,r

                sub r5,r5,r8   ;r5=n-r

                mov r0,r5

                bl fact           ; (N-R)! IS FOUND

                mov r5,r6  ;r5=(n-r)!

  mul r9,r5,r1 ;r5=r!\*(n-r)!

                bl div

stop          b stop

fact            EOR R6,R6,R6

                MOV R6,R0

MOV R4,R6

LOOP SUBS R4,R4,#1

MULNE R7,R6,R4

MOV R6,R7

BNE LOOP

                mov PC,LR

  div             mov r0,r10

                mov r1,r9

                mov r3,#0

loop

cmp r1,#0

beq err

cmp r0,r1

blt done

add r3,r3,#1

sub r0,r0,r1

b loop

err

mov r3,#0xffffffff

done

        ldr r2,=remain

        ldr r4,=quotient

str r0,[r2]

str r3,[r4]

        movpc,lr

 END

**Experiment 6: Search a key element “X” in a list of „n‟ 16-bit numbers using binary search algorithm**

|  |
| --- |
| SRAM\_BASE EQU 0x40000000 |
| LDR sp, =SRAM\_BASE |
| LDR r3, =SRAM\_BASE + 200 |
|  |
|  |
|  |
| NUM EQU 11 |
| SIZE EQU 1 |
|  |
| ADR r6, array |
| MOV r1, #0 //lb |
| MOV r2, #NUM - 1 //ub |
| MOV r5, #17 // Enter desired search value here |
| STMDB r3!, {r6,r1,r2,r5, r0} |
|  |
|  |
|  |
| main |
| BL recursiveBinarySearch |
| B main |
|  |
|  |
| recursiveBinarySearch |
| STMDB sp!, {r4,r7,r8,r9,r10,r11,r12,lr} // preserving all the registers we have to work with other than the four that were passed by reference |
| LDMFD r3!, {r11,r7,r8,r10, r0} // r7 = r1, r8 = r2, r10 = r5, r11 = r6 |
| CMP r7, r8//r1 > r2? |
| BGT stop |
|  |
| ADD r9, r7, r8 //first + last |
| MOV r9, r9, ASR #1//(first + last) / 2 |
| LDR r12, [r11, r9,LSL #2] //load the entry |
| ADD r11, r9, LSL #2 |
| MOV r0, r11 // keep this address ( r11 will be reset later ) |
| ADD r4, r9, LSL #1 // r4 = scratch |
| CMP r12, r10//a[half] > value ? |
| SUBGT r8, r9, #1//ub = half - 1 |
| ADDLE r7, r9, #1 //lb = half + 1 |
| LDR r11, [r3, #-4] // reset r11 |
| STMFD r3!, {r11,r7,r8,r10, r0} // main registers we're using |
| LDMIA sp!, {r4,r7,r8,r9,r10,r11,r12,pc} // preserve |
|  |
| MOV pc, lr |
|  |
| Stop B stop |
|  |
|  |
| array DCD 3,6,8,12,17,22,45,67,99,2089,30001 |
| END |

**Experiment 7: C Programs for matrix multiplication, matrix addition and sparse matrix implementation.**

**//multiplication**

#include <stdio.h>

int main()

{

int m, n, p, q, c, d, k, sum = 0;

int first[10][10], second[10][10], multiply[10][10];

printf("Enter number of rows and columns of first matrix\n");

scanf("%d%d", &m, &n);

printf("Enter elements of first matrix\n");

for (c = 0; c < m; c++)

for (d = 0; d < n; d++)

scanf("%d", &first[c][d]);

printf("Enter number of rows and columns of second matrix\n");

scanf("%d%d", &p, &q);

if (n != p)

printf("The matrices can't be multiplied with each other.\n");

else

{

printf("Enter elements of second matrix\n");

for (c = 0; c < p; c++)

for (d = 0; d < q; d++)

scanf("%d", &second[c][d]);

for (c = 0; c < m; c++) {

for (d = 0; d < q; d++) {

for (k = 0; k < p; k++) {

sum = sum + first[c][k]\*second[k][d];

}

multiply[c][d] = sum;

sum = 0;

}

}

printf("Product of the matrices:\n");

for (c = 0; c < m; c++) {

for (d = 0; d < q; d++)

printf("%d\t", multiply[c][d]);

printf("\n");

}

}

return 0;

}

**// sparse matrix**

#include <stdio.h>

int main()

{

// Assume 4x5 sparse matrix

int sparseMatrix[4][5] =

{

{0 , 0 , 3 , 0 , 4 },

{0 , 0 , 5 , 7 , 0 },

{0 , 0 , 0 , 0 , 0 },

{0 , 2 , 6 , 0 , 0 }

};

int compactMatrix[10][10];

int size = 0;

for (int i = 0; i < 4; i++)

for (int j = 0; j < 5; j++)

if (sparseMatrix[i][j] != 0)

size++;

// number of columns in compactMatrix (size) must be

// equal to number of non - zero elements in

// sparseMatrix

//int compactMatrix[3][size];

// Making of new matrix

int k = 0;

for (int i = 0; i < 4; i++)

for (int j = 0; j < 5; j++)

if (sparseMatrix[i][j] != 0)

{

compactMatrix[0][k] = i;

compactMatrix[1][k] = j;

compactMatrix[2][k] = sparseMatrix[i][j];

k++;

}

for (int i=0; i<3; i++)

{

for (int j=0; j<size; j++)

printf("%d ", compactMatrix[i][j]);

printf("\n");

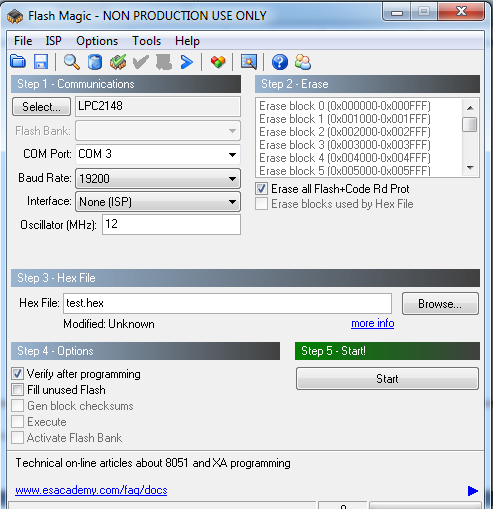
}

return 0;

}

**Assignment: Write a C program to perform addition and subtraction of matrices.**

**Steps for hardware Interfacing using Flashmagic**

****

**Experiment 8: Familiarize I/O ports of LPC 2148 – on/off control of LEDs using switches.**

void Delay(unsigned int dms);

unsigned int delay\_ms,ledval,n; // Define all your variables here

void Delay(unsigned int dms)

{

delay\_ms = dms;

while(delay\_ms > 0)

{

delay\_ms--;

}

}

void Init\_GPIO(void)

{

PINSEL0 = 0x00;

PINSEL1 = 0x00;

g\_pGPIO0->IODIR = 0XFFFFFFFF;

g\_pGPIO1->IODIR = 0XFFFFFFFF;

}

main()

{

Init\_GPIO();

while(1)

{ for(n = 0;n<8 ;n++)

{switch(n)

{case 0: ledval = 0x01010101;

break;

case 1: ledval = 0x02020202;

break;

case 2: ledval = 0x04040404;

break;

case 3: ledval = 0x08080808;

break;

case 4: ledval = 0x10101010;

break;

case 5: ledval = 0x20202020;

break;

case 6: ledval = 0x40404040;

break;

case 7: ledval = 0x80808080;

break;

} }

**Assignment: Binary Counter using LEDs**

**Experiment 9: Interface keypad and display the key pressed on LCD.**

// Initial C Source File

#include "xstdsys.h"

//Please note that you have to connect H1 to LCD data and H2 to LCD control

// Define all your variables here

unsigned int cmd8[] = {0X38,0x38,0x0E,0x02,0x01,0x00};

unsigned int msg[] = {'H','e','l','l','o',0x20,'A','D','M',0x20,0x00};

unsignedintlcdval,index,delay\_ms;

// define prototypes

voidInitLPC(void);

void Delay(unsigned int);

voidInitLPC(void)

{

// use the Wizards (see Wizards menu) to configure the on-chip peripherals

PINSEL0 = 0x00L;

g\_pGPIO0->IODIR = 0XFFFFFFFF;

}

void Delay(unsigned intdms)

{

// the timer has to work for this loop to function

delay\_ms = dms;

while(delay\_ms> 0)

{

delay\_ms--;

}

}

voidInitLCD()

{

index=0;

lcdval=cmd8[index];

while(lcdval !=0x0)

{

g\_pGPIO0->IOSET = lcdval;

//Toggle E pin

lcdval |= 0x400;

g\_pGPIO0->IOSET = lcdval;

Delay(500);

g\_pGPIO0->IOCLR=0xFFFF;

index++;

lcdval=cmd8[index];

}

}

voidShowMsg()

{

index=0;

lcdval=msg[index];

while(lcdval !=0x0)

{

g\_pGPIO0->IOSET = lcdval;

//Toggle E pin

lcdval |= 0x500;

g\_pGPIO0->IOSET = lcdval;

Delay(500);

g\_pGPIO0->IOCLR=0xFFFF;

index++;

lcdval=msg[index];

}

}

void main(void)

{

InitLPC();

// use the Wizards (see Wizards menu) to configure the on-chip peripherals

index=0;

while(1)

{

InitLCD();

ShowMsg();

Delay(5000);

}

}

**Assignment: Simulate calculator and display the result.**

**Experiment 10: Waveform generation using the internal DAC of LPC 2148.**

//Sine

#include "lpc214x.h"

#include "stdint.h"

voiddelay\_ms(uint16\_t j)

{

uint16\_tx,i;

for(i=0;i<j;i++)

{

for(x=0; x<6000; x++); /\* loop to generate 1 milisecond delay with Cclk = 60MHz \*/

}

}

int main (void)

{

uint16\_t value;

uint16\_ti = 0;

uint16\_tsintable[64]= {512,562,611,660,707,753,796,836,

873,907,937,963,984,1001,1013,1021,

1023,1021,1013,1001,984,963,937,907,

873,836,796,753,707,660,611,562,

512,461,412,363,316,270,227,187,

150,116,86,60,39,22,10,2,

0,2,10,22,39,60,86,116,

150,187,227,270,316,363,412,461 };

PINSEL1 = 0x00080000; /\* P0.25 as DAC output \*/

IO0DIR = 0xFFFFFFFF; /\* Input pins for switch. P0.8 sine, P0.9 triangular, P0.10 sawtooth, P0.11 square \*/

while(1){

while(i<64)

{

value=(sintable[i]\*50);

DACR=value;

delay\_ms(1);

i++;

}

i=0;

}

}

//triangle

#include "lpc214x.h"

#include "stdint.h"

voiddelay\_ms(uint16\_t j)

{

uint16\_tx,i;

for(i=0;i<j;i++)

{

for(x=0; x<6000; x++); /\* loop to generate 1 milisecond delay with Cclk = 60MHz \*/

}

}

int main (void)

{

uint16\_t value;

uint16\_ti = 0;

PINSEL1 = 0x00080000; /\* P0.25 as DAC output \*/

IO0DIR = 0xFFFFFFFF; /\* Input pins for switch. P0.8 sine, P0.9 triangular, P0.10 sawtooth, P0.11 square \*/

while(1)

{

i=0;

while(i!=1023)

{

DACR=i\*50;

i++;

}

i=1023;

while(i!=0)

{

DACR=i\*50;

i--;

}

}

}

//SQUARE

#include "lpc214x.h"

#include "stdint.h"

voiddelay\_ms(uint16\_t j)

{

uint16\_tx,i;

for(i=0;i<j;i++)

{

for(x=0; x<6000; x++); /\* loop to generate 1 milisecond delay with Cclk = 60MHz \*/

}

}

int main (void)

{

uint16\_t value;

uint16\_ti = 0;

PINSEL1 = 0x00080000; /\* P0.25 as DAC output \*/

IO0DIR = 0xFFFFFFFF; /\* Input pins for switch. P0.8 sine, P0.9 triangular, P0.10 sawtooth, P0.11 square \*/

while(1)

{

DACR=1023\*50;

delay\_ms(10);

DACR=0;

delay\_ms(10);

}

}

**Assignment: Generate Sawtooth waveform.**

**Experiment 11: Convert a given analog voltage to digital using ADC of LPC 2148.**

// Initial C Source File

#include "xstdsys.h"

unsigned char adcval;

//Connect Channel P0.30 to Potentiometer near Relay

//Connect P0.0 to LEDs and monitor working

void init\_adc0()

{

// Force pin 0.30 to function as AD0.3

g\_pPINCONNECT->PINSEL1 = 0X10000000;

// Default the power is enabled in PCONP register

g\_pSYSTEM->PCONP |= PCAD0;

// Initialise ADC converter for 8 bit mode

// Remove from PDN Mode, Select CLKS conversion rate, use ADC0 ch:3

g\_pADC0->CR = PDN | CLKS1 | SEL3;

}

void read\_adc()

{

int ival;

// Manually Start Conversion

g\_pADC0->CR |= START0;

// Wait for the conversion to complete

while (!(g\_pADC0->DR3 & DONE));

// Read ADC data register

g\_pGPIO0->IOCLR = adcval;

ival = g\_pADC0->DR3;

ival = ((ival >> 6) & 0x3FF); //Define preceision

ival >>=2;

adcval = (unsigned char) ival;

}

main()

{

// use the Wizards (see Wizards menu) to configure the on-chip peripherals

g\_pGPIO0->IODIR = 0XFF;

init\_adc0();

while (1)

{

read\_adc();

g\_pGPIO0->IOSET = adcval;

}

}

**Assignment: Convert square wave to its digital equivalent.**

**Experiment 12: Design and display a 4 digit counter.**

#include "lpc214x.h"

#include "stdint.h"

#define IO1 0x10000

#define IO2 0x20000

#define IO3 0x40000

#define IO4 0x80000

#define IOX 0xF0000

#define IOXcl 0xFFFFF

//Multiplexed 7segment Display

int count=0x0000;

unsigned int d0,d1,d2,d3;

unsigned char seg[] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x67,0x77,0x7c,0x39,0x5e,0x79,0x71,0x00};

voidinit\_gpio()

{

PINSEL0 = 0x00000000;

PINSEL1 = 0x00000000;

PINSEL2 = 0x00000000;

IO0DIR = 0XFFFFFFFF;

IO1DIR = 0XFFFFFFFF;

}

void delay()

{

int c = 100000;

while(c) //while count is more than zero loop

{

c--;

}

}

voidshow\_disp()

{

//Digit 3

d3 = count & 0x0F000;

d3 >>= 12;

IO0CLR = IOXcl;

IO0SET= seg[d3]; //Willdisplay data 1 on 7seg

IO1SET = IOX; //ALL display are OFF

IO1CLR = IO4; //Display1 is made on

delay();

IO1SET = IOX; //ALL display are OFF

//Digit 2

d2 = count & 0x0F00;

d2 >>= 8;

IO0CLR = IOXcl;

IO0SET= seg[d2]; //Willdisplay data 2 on 7seg

IO1SET = IOX; //ALL display are OFF

IO1CLR = IO3; //Display1 is made on

delay();

IO1SET= IOX; //ALL display are OFF

//Digit 1

d1 = count & 0x00F0;

d1 >>= 4;

IO0CLR = IOXcl;

IO0SET = seg[d1]; //Willdisplay data 3 on 7seg

IO1SET = IOX; //ALL display are OFF

IO1CLR = IO2; //Display1 is made on

delay();

IO1SET = IOX; //ALL display are OFF

//Digit 0

d0 = count & 0x000F;

IO0CLR = IOXcl;

IO0SET = seg[d0]; //Will display data 4 on 7seg

IO1SET = IOX; //ALL display are OFF

IO1CLR = IO1; //Display1 is made on

delay();

IO1SET = IOX; //ALL display are OFF

}

int main( void )

{

init\_gpio();

while(1)

{

show\_disp();

count++;

count&= 0xFFFF;

}

}

**Question Bank**

1. a) Write an assembly language program to perform the two, 64 bit data addition and subtraction using multiple register load/store instruction.

b) Interface a 4x4, keypad to LPC2148 microcontroller, if key is pressed display it on 7

Segment.

1. a) Write an ALP to arrange the given 32 bit, ‘N’ numbers in ascending order using bubble sort technique.

b) Interface LCD to LPC2148 microcontroller display the given message in single line.

1. a) Write an ALP to compute nCr, where ‘n’ and ‘r’ are positive numbers.

b) Interface 4, 7-segment display to LPC2148 microcontroller modules to count from

0000-FFFF.

1. a) Write an ALP to check palindrome of a given string.

b) Interface DAC module to LPC2148 microcontroller to generate Triangular and square

waveforms.

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

b) Interface DAC module to LPC2148 microcontroller to generate Sine and square

waveforms.

1. a) Write an ALP to convert the given hexadecimal number into packed BCD form.

b) Interface 7-segment display module to LPC2148 microcontroller to count from

0-F.

1. a) Write an ALP to convert the given unpacked BCD number to hex form.

b) Interface 8 LEDs to LPC2148 microcontroller blink them with suitable time delay

1. a) Write an ALP in THUMB mode to compute the given series 12 + 22 + 32+42+…..+102

b) Interface LCD to LPC2148 microcontroller display the given message in double line.

1. a) Write an ALP in THUMB mode to find the largest and smallest number in a given

array of data.

b) Interface a 4x4, keypad to LPC2148 microcontroller, if key is pressed display it on 7

Segment.

1. a) Write an ALP to arrange the given 32 bit, ‘N’ numbers in descending order using

bubble sort technique.

b) Interface DAC module to LPC2148 microcontroller to generate Triangular and

Sawtooth waveforms.

1. a) Write an ALP in THUMB mode to find the Factorial of a given number.

b) Interface 4, 7-segment display to LPC2148 microcontroller module to count from

0000-FFFF.

1. a) Write an ALP to copy a given string from source to destination memory location and

reverse it.

b) Write a C progeam to convert a given analog voltage to digital using ADC of LPC 2148.