|  |  |  |  |
| --- | --- | --- | --- |
| Microcontrollers | | | |
| Course Code: | BCS402 | CIE Marks | 50 |
| Teaching Hours/Week (L:T:P:S) | 3:0:2:0 | SEE Marks | 50 |
| Total Hours of Pedagogy | 40 T + 20 P | Total Marks | 100 |
| Credits | 04 | Exam Hours | 03 |

|  |
| --- |
| Course Learning Objectives: |
| CLO 1: : Understand the fundamentals of ARM-based systems and basic architecture of CISC and RISC..CLO 2: Familiarize with ARM programming modules along with registers, CPSR and Flags..CLO 3: : Develop ALP using various instructions to program the ARM controller.CLO 4: Understand the Exceptions and Interrupt handling mechanism in Microcontrollers..CLO 5 : Discuss the ARM Firmware packages and Cache memory polices. |

|  |
| --- |
| Course outcome (Course Skill Set) |
| At the end of the course, the student will be able to:CO 1. Explain the ARM Architectural features and Instructions.CO 2. Develop programs using ARM instruction set for an ARM MicrocontrollerCO 3. Explain C-Compiler Optimizations and portability issues in ARM Microcontroller.CO 4. Apply the concepts of Exceptions and Interrupt handling mechanisms in developing applications.CO 5. Demonstrate the role of Cache management and Firmware in Microcontrollers. |

|  |  |  |
| --- | --- | --- |
| Laboratory Component: | | |
| Module 1: ARM Processor Fundamentals | | |
| Sl. No | Name | Page no |
| 1 | Using Keil software, observe the various Registers, Dump, CPSR, with a simple Assembly Language Programs (ALP). |  |
| Module 2: Introduction to the ARM Instruction Set | | |
| 2 | Develop and simulate ARM ALP for Data Transfer, Arithmetic and Logical operations (Demonstrate with the help of a suitable program). |  |
| 3 | Develop an ALP to multiply two 16-bit binary numbers. |  |
| 4 | Develop an ALP to find the sum of first 10 integer numbers. |  |
| 5 | Develop an ALP to find the largest/smallest number in an array of 32 numbers. |  |
| 6 | Develop an ALP to count the number of ones and zeros in two consecutive memory locations |  |
| Module 3: C Compilers and Optimization, ARM programming using Assembly language | | |
| 7 | Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort |  |
| 8 | Simulate a program in C for ARM microcontroller to find factorial of a number |  |
| 9 | Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase. |  |
| Module 4& 5 Embedded System Components: | | |
| 10 | Demonstrate enabling and disabling of Interrupts in ARM. |  |
| 11 | Demonstrate the handling of divide by zero, Invalid Operation and Overflow exceptions in ARM. |  |

|  |  |  |
| --- | --- | --- |
| ADDITIONAL EXPERIMENTS | | |
| 1 | ALP to determine the given 16 bit number is ODD or EVEN. |  |
| 2 | Interface a simple Switch and display its status through Relay, Buzzer and LED. |  |

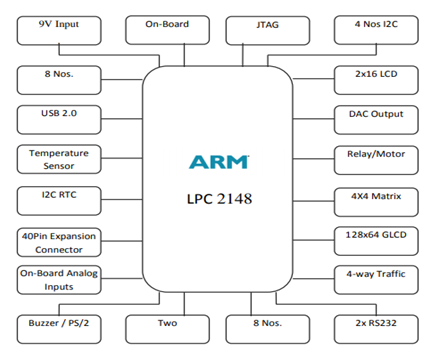
|  |
| --- |
| Conduct of Practical Examination: |
| Practical Sessions need to be assessed by appropriate rubrics and viva-voce method. This will contribute to 20 marks.• Rubrics for each Experiment taken average for all Lab components – 15 Marks.• Viva-Voce– 5 Marks (more emphasized on demonstration topics) |

# Introduction to ARM7

TheLPC2141/42/44/46/48microcontrollersarebasedona16-bit/32-bitARM7TDMI-SCPUwith real-time emulation and embedded trace support, that combine microcontroller with embedded high-speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For criticalcodesizeapplications,thealternative16-bitThumbmodereducescodebymorethan30

% with minimal performance penalty. Due to their tiny size and low power consumption, LPC2141/42/44/46/48 are ideal for applications where miniaturization is a key requirement, such asaccesscontrolandpoint-ofsale.SerialcommunicationsinterfacesrangingfromaUSB2.0Full- speed device, multiple UARTs, SPI, SSP to I2C-bus and on-chip SRAM of 8 kB up to 40 kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low-end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channelsand45fastGPIOlineswithuptonineedgeorlevelsensitiveexternalinterruptpinsmake these microcontrollers suitable for industrial control and medical systems.

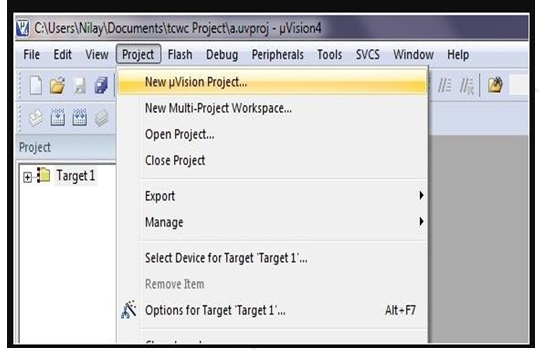
**BLOCK Diagram of ARMLPC2148**



**Step 1:** After opening Keil uV4, Go to Project tab and click on close project

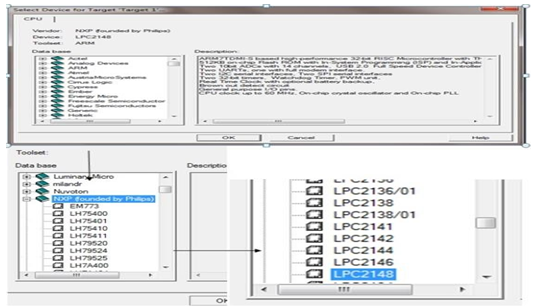
Then Create new uVision project

Now Select new folder and give name to Project.

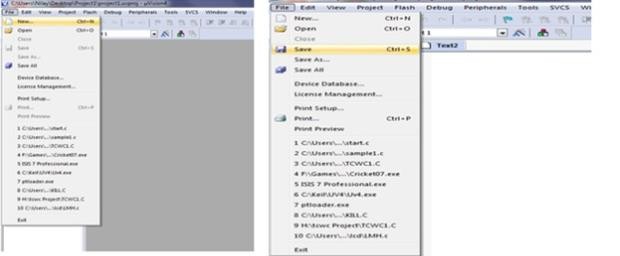
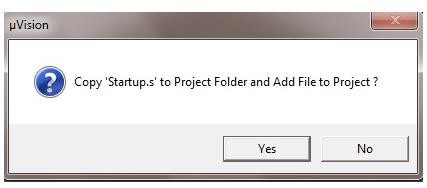
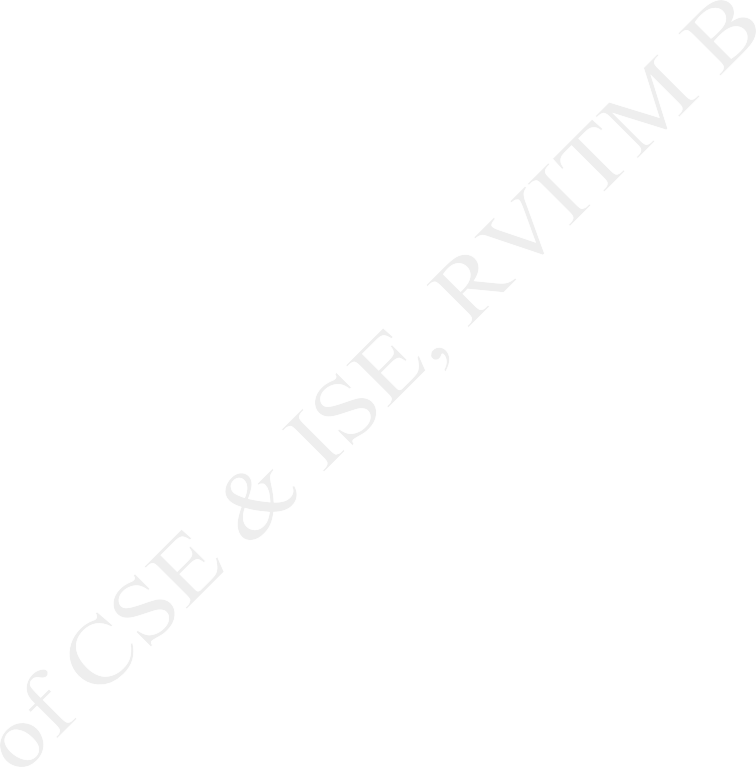


**Step 2:** After Creating project now Select your device model. Example.NXP-LPC2148

[You can change it later from project window.]



**Step 3:**So now your project is created and **Message** window will appear to add start up file of your Device click on **Yes** so it will be added to your project folder

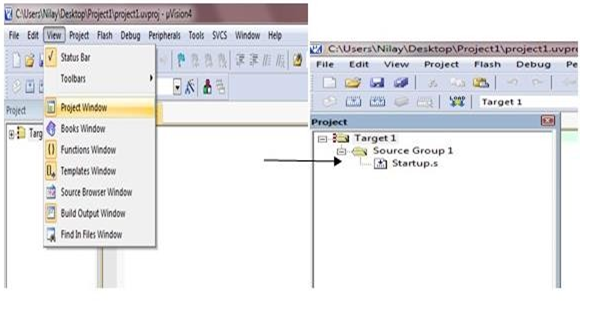


**Step4**:Now go to File and create new file and save it with **.C** extension if you will write program

in C language or save with **.asm** for **assembly** language.

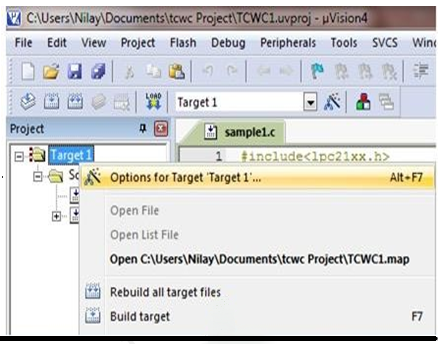
**Step5**:Now write your program and save it again.

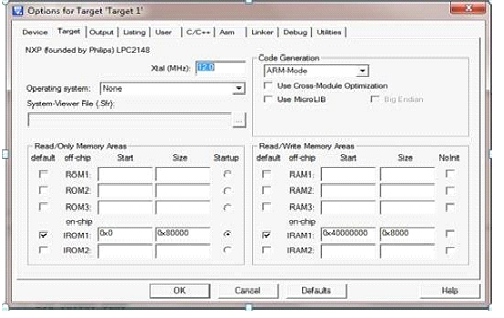
**Step6**: After that on left you see project window [if it’s not there….go to View tab and click on project window.



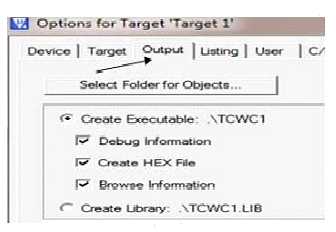
Right Click on Target and click on options for target.

Here you can change your device also

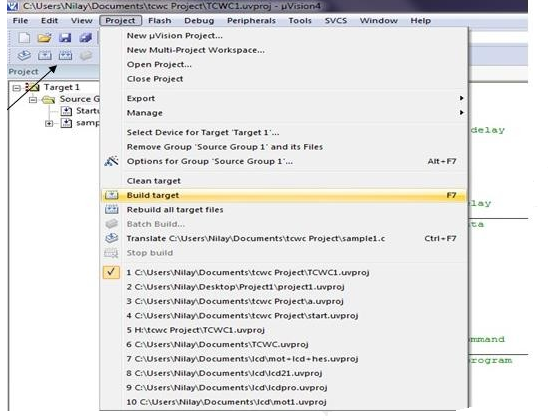




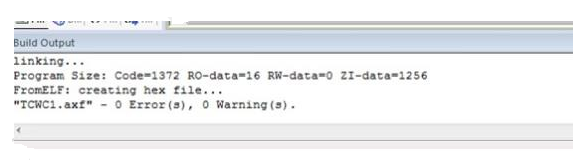
Click output tab here and check the HEX files, if you are generating hex files



Step 7: Now expand Target and you will see the Source Group

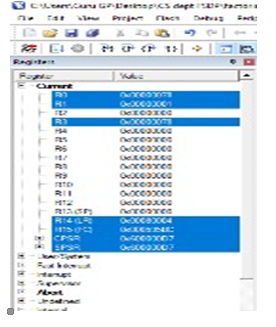


Step 8: Build Target and Run the program if no errors



|  |
| --- |
| Module1: ARM Processor Fundamentals |
| Program 1. Using Keil software, observe the various registers, dump, CPSR, with a simple ALP program AREA MULTIPLY, CODE, READONLY  ENTRY ; Mark first instruction to execute  START  MOV R1, #6400 ; STORE FIRST NUMBER IN R1  MOV R2, #3200 ; STORE SECOND NUMBER IN R2  MUL R3, R1, R2 ; MULTIPLICATION  NOP  NOP  END ; Mark end of file  **Output:**  1st Input : Register R1 =6400  2nd Input : Register R2 =3200  Result : Register R3=13880000 |

|  |
| --- |
| Module2: Introduction to the ARM Instruction Set |
| Program 2: Write a program to find the sum of the first 10 integer numbers.AREA SUM, CODE, READONLYENTRYMOV R1, #10 ; load 10 to registerMOV R2, #0 ; empty the register to store resultLOOPADD R2, R2, R1 ; add the content of R1 with result at R2SUBS R1, #0x01 ; Decrement R1 by 1BNE LOOP ; repeat till R1 goes 0BACK B BACK ; jumps back to C codeENDOutput:Result can be viewed in Register R2 in hex decimal valuesProgram3: Write a program to find factorial of a number. AREA FACTORIAL, CODE, READONLY  ENTRY ; Mark first instruction to execute  START  MOV R0, #7 ; STORE FACTORIAL NUMBER IN R0  MOV R1, R0 ; MOVE THE SAME NUMBER IN R1  FACT SUBS R1, R1, #1 ; SUBTRACTION  CMP R1, #1 ; COMPARISON  BEQ STOP  MUL R3, R0, R1 ; MULTIPLICATION  MOV R0, R3 ; Result  BNE FACT ; BRANCH TO THE LOOP IF NOT EQUAL  STOP  NOP  NOP  NOP  END ; Mark end of file  **Output:**  Result can be viewed in Register R0 |



**Program 4. Write a program to add an array of 16 bit numbers and store the 32 bit result in internal RAM .**

AREA ADDITION, CODE, READONLY

ENTRY ; Mark first instruction to execute

START

MOV R5, #6 ; INTIALISE COUNTER TO 6(i.e. N=6)

MOV R0, #0 ; INTIALISE SUM TO ZERO

LDR R1, =VALUE1 ; LOADS THE ADDRESS OF 1ST VALUE

LOOP

LDRH R3, [R1], #02 ; READ 16 BIT DATA

ADD R0, R0, R3 ; ADD R0=R0+R3

SUBS R5, R5, #1 ; DECREMENT COUNTER

CMP R5, #0

BNE LOOP ; LOOK BACK TILL ARRAY ENDS

LDR R4, =RESULT ; LOADS THE ADDRESS OF RESULT

STR R0, [R4] ; STORES THE RESULT IN R0

JMP B JMP

VALUE1 DCW 0X1111,0X2222,0X3333,0XAAAA,0XBBBB,0XCCCC

; ARRAY OF 16 BIT NUMBERS (N=6)

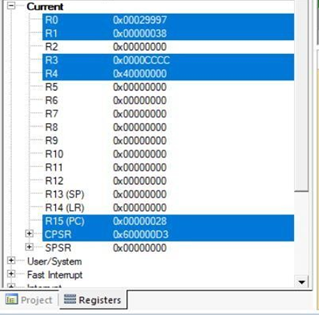
AREA DATA2, DATA, READWRITE ; TO STORE RESULT IN GIVEN ADDRESS

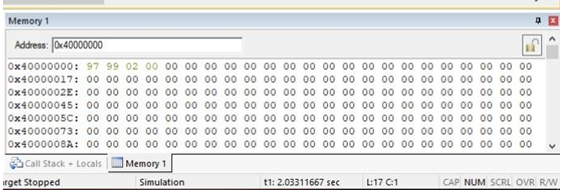
RESULT DCD 0X0

END ; Mark end of file

Output:

Result can be viewed in Memory location address specified in R4 and also in register R0





# Program 5: Write a program to find the square of a number (1 to 10) using look-up table.



AREA SQUARE, CODE, READONLY

ENTRY ; Mark first instruction to execute

START

LDR R0, = TABLE1 ; Load start address of Lookup table

LDR R2, = 0X40000000

LDR R1, [R2] ; Load no whose square is to be find

MOV R1, R1, LSL #0x2

; Generate address corresponding to square of given no

ADD R0, R0, R1 ; Load address of element in Lookup table

LDR R3, [R0] ; Get square of given no in R3

NOP

NOP

NOP

;Lookup table contains Squares of no s from 0 to 10 (in hex)

TABLE1 DCD 0X00000000 ;SQUARE OF 0=0

DCD 0X00000001 ;SQUARE OF 1=1

DCD 0X00000004 ;SQUARE OF 2=4

DCD 0X00000009 ;SQUARE OF 3=9

DCD 0X00000010 ;SQUARE OF 4=16

DCD 0X00000019 ;SQUARE OF 5=25

DCD 0X00000024 ;SQUARE OF 6=36

DCD 0X00000031 ;SQUARE OF 7=49

DCD 0X00000040 ;SQUARE OF 8=64

DCD 0X00000051 ;SQUARE OF 9=81

DCD 0X00000064 ;SQUARE OF 10=100

END ; Mark end of file

**Output:**

Enter Input number in memory location specified in Register R2

Result can be viewed in Register R3

# 

# Program6a: Write a program to find the largest/smallest number in an array of 32 numbers.

AREA LARGEST, CODE, READONLY

ENTRY ; Mark first instruction to execute

START

MOV R5, #6 ; INTIALISE COUNTER TO 6(i.e. N=7)

LDR R1, =VALUE1 ; LOADS THE ADDRESS OF FIRST VALUE

LDR R2, [R1], #4 ; WORD ALIGN T0 ARRAY ELEMENT

LOOP

LDR R4, [R1], #4 ; WORD ALIGN T0 ARRAY ELEMENT

CMP R2, R4 ; COMPARE NUMBERS

BHI LOOP1 ; IF THE 1st NUMBER IS > THEN GOTO LOOP1

MOV R2, R4 ; IF THE 1st NUMBER IS < THEN MOV

; CONTENT R4 TO R2

LOOP1

SUBS R5, R5, #1 ; DECREMENT COUNTER

CMP R5, #0 ; COMPARE COUNTER TO 0

BNE LOOP ; LOOP BACK TILL ARRAY ENDS

LDR R4, =RESULT ; LOADS THE ADDRESS OF RESULT

STR R2, [R4] ; STORES THE RESULT IN R2

NOP

back B back ; ARRAY OF 32 BIT NUMBERS (N=7)

VALUE1

DCD 0X44444444

DCD 0X22222222

DCD 0X11111111

DCD 0X33333333

DCD 0XAAAAAAAA

DCD 0X88888888

DCD 0X99999999

AREA DATA2, DATA, READWRITE

RESULT DCD 0X0 ; ; TO STORE RESULT IN GIVEN ADDRESS

END ; Mark end of file

Output:

Result can be viewed in Memory location address specified in R4 and also in register R2

**Experiment No 6b. Write a program to find the smallest number in an array of 32 numbers.**

AREA SMALLEST, CODE, READONLY

ENTRY ; Mark first instruction to execute

START

MOV R5, #6 ; INTIALISE COUNTER TO 6(i.e. N=7)

LDR R1, =VALUE1 ; LOADS THE ADDRESS OF FIRST VALUE

LDR R2, [R1], #4 ; WORD ALIGN T0 ARRAY ELEMENT

LOOP

LDR R4, [R1], #4 ; WORD ALIGN T0 ARRAY ELEMENT

CMP R2, R4 ; COMPARE NUMBERS

BLS LOOP1 ; IF THE 1st NUMBER IS < THEN GOTO LOOP1

MOV R2, R4 ; IF THE 1st NUMBER IS > THEN MOV

; CONTENT R4 TO R2

LOOP1

SUBS R5, R5, #1 ; DECREMENT COUNTER

CMP R5, #0 ; COMPARE COUNTER TO 0

BNE LOOP ; LOOP BACK TILL ARRAY ENDS

LDR R4, =RESULT ; LOADS THE ADDRESS OF RESULT

STR R2, [R4] ; STORES THE RESULT IN R1

NOP

NOP

NOP

; ARRAY OF 32 BIT NUMBERS (N=7)

VALUE1

DCD 0X44444444

DCD 0X22222222

DCD 0X11111111

DCD 0X22222222

DCD 0XAAAAAAAA

DCD 0X88888888

DCD 0X99999999

AREA DATA2, DATA, READWRITE

; TO STORE RESULT IN GIVEN ADDRESS

RESULT DCD 0X0

END ; Mark end of file

**Output:**

Result can be viewed in Memory location address specified in R4 and also in register R2

# Program7:Write a program to arrange a series of 32-bit numbers in ascending order.

AREA ASCENDING, CODE, READONLY

ENTRY ; Mark first instruction to execute

START

MOV R8, #4 ; INTIALISE COUNTER TO 4 (i.e. N=4)

LDR R2, =CVALUE ; ADDRESS OF CODE REGION

LDR R3, =DVALUE ; ADDRESS OF DATA REGION

LOOP0

LDR R1, [R2], #4 ; LOADING VALUES FROM CODE REGION

STR R1, [R3], #4 ; STORING VALUES TO DATA REGION

SUBS R8, R8, #1 ; DECREMENT COUNTER

CMP R8, #0 ; COMPARE COUNTER TO 0

BNE LOOP0 ; LOOP BACK TILL ARRAY ENDS

START1 MOV R5, #3 ; INTIALISE COUNTER TO 3(i.e. N=4)

MOV R7, #0 ; FLAG TO DENOTE EXCHANGE HAS OCCURED

LDR R1, =DVALUE ; LOADS THE ADDRESS OF 1st VALUE

LOOP LDR R2, [R1], #4 ; WORD ALIGN T0 ARRAY ELEMENT

LDR R3, [R1] ; LOAD SECOND NUMBER

CMP R2, R3 ; COMPARE NUMBERS

BLT LOOP2 ; IF THE 1st NUMBER IS < THEN GOTO LOOP2

STR R2, [R1], #-4

STR R3, [R1]

MOV R7, #1 ; FLAG DENOTING EXCHANGE HAS TAKEN PLACE

ADD R1, #4 ; RESTORE THE PTR

LOOP2

SUBS R5, R5, #1 ; DECREMENT COUNTER

CMP R5, #0 ; COMPARE COUNTER TO 0

BNE LOOP ; LOOP BACK TILL ARRAY ENDS

CMP R7, #0 ; COMPARING FLAG

BNE START1

; IF FLAG IS NOT ZERO THEN GO TO START1 LOOP

NOP

NOP

NOP

; ARRAY OF 32 BIT NUMBERS (N=4) IN CODE REGION

CVALUE

DCD 0X44444444

DCD 0X11111111

DCD 0X33333333

DCD 0X22222222

AREA DATA1, DATA, READWRITE

; ARRAY OF 32 BIT NUMBERS IN DATA REGION

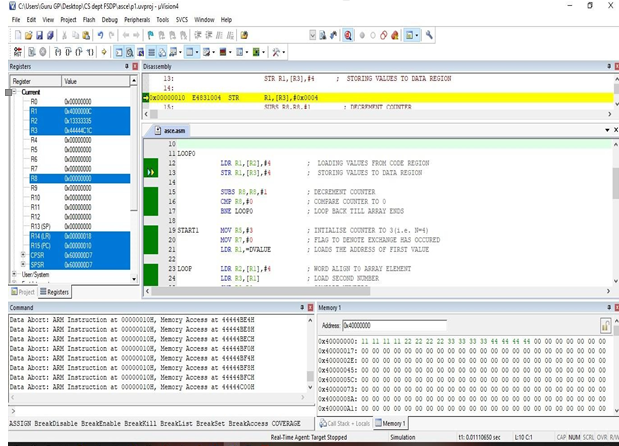
DVALUE

DCD 0X00000000

END ; Mark end of file

**Output:**

Result can be viewed at location DVALUE (stored in R3)



**Program 8: Write a program to count the number of one s and zeros in two consecutive memory locations**.

AREA ONEZERO , CODE, READONLY

ENTRY ; Mark first instruction to execute

START

MOV R2, #0 ; COUNTER FOR ONES

MOV R3, #0 ; COUNTER FOR ZEROS

MOV R7, #2 ; COUNTER TO GET TWO WORDS

LDR R6, =VALUE ; LOADS THE ADDRESS OF VALUE

LOOP MOV R1, #32 ; 32 BITS COUNTER

LDR R0, [R6], #4 ; GET THE 32 BIT VALUE

LOOP0 MOVS R0, R0, ROR #1 ; RIGHT SHIFT TO CHECK CARRY BIT (1's/0's)

BHI ONES

; IF CARRY BIT IS 1 GOTO ONES BRANCH OTHERWISE NEXT

ZEROS ADD R3, R3, #1

; IF CARRY BIT IS 0 THEN INCREMENT THE COUNTER BY 1(R3)

B LOOP1 ; BRANCH TO LOOP1

ONES ADD R2, R2, #1

; IF CARRY BIT IS 1 THEN INCREMENT THE COUNTER

BY 1(R2)

LOOP1 SUBS R1, R1, #1 ; COUNTER VALUE DECREMENTED BY 1

BNE LOOP0 ; IF NOT EQUAL GOTO TO LOOP0 CHECKS 32BIT

SUBS R7, R7, #1 ; COUNTER VALUE DECREMENTED BY 1

CMP R7, #0 ; COMPARE COUNTER R7 TO 0

BNE LOOP ; IF NOT EQUAL GOTO TO LOOP

NOP

NOP

NOP

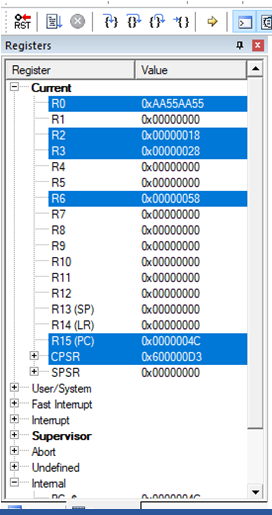
JMP B JMP

VALUE DCD 0X11111111, 0XAA55AA55 ; TWO VALUES IN AN ARRAY

END ; Mark end of file

**Output:**

Result in R2 for ONES & R3 for ZEROS



**Program9:Display“HelloWorld”messageusingInternalUART**.

#include<lpc214x.h>

void uart\_init(void);

unsigned int delay;

unsigned char \*ptr;

unsigned char arr[]="HELLO WORLD\r";

int main ()

{

while (1)

{

uart\_init();

ptr = arr;

while (\*ptr! ='\0')

{

U0THR=\*ptr++;

while (! (U0LSR & 0x40) == 0x40);

for (delay=0; delay<=600; delay++);

}

for (delay=0; delay<=60000; delay++);

}

}

void uart\_init(void)

{

PINSEL0=0X0000005; //select TXD0 and RXD0

lines U0LCR = 0X00000083; //enable baud rate divisor

loading U0DLM = 0X00; //select the data format

U0DLL = 0x13; //select baud rate 9600

bps U0LCR = 0X00000003;

}

**Result: Hello World will be displayed on terminal.**

.