MICROCONTROLLER AND EMBEDDED SYSTEMS LABORATORY

MODULE -1

1. **Using keil software observes the various Registers, Dump. CPSR with a single Assembly Language Programs (ALP)?**

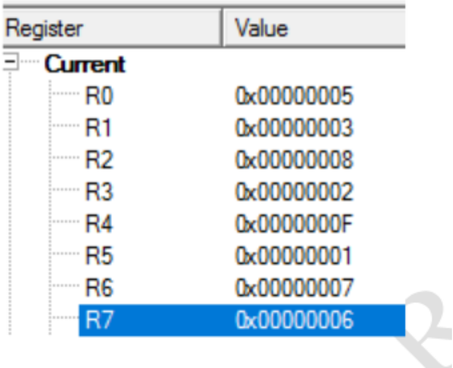
PROJECT CREATION IN KEILUV4 IDE:
Create a project folder before creating NEW project.
• Open Keil uVision4 IDE software by double clicking on "Keil Uvision4" icon.
• Go to "Project" then to "New uVision Project" and save it with a name in the respective project folder, already you created.
• Select the device as "NXP" In that "LPC2148" then press OK and then press "YES" Button to add
"startup.s" file.
• In startup file go to Configuration Wizard. In Configuration Wizard window uncheck PLL Setup
and check VPBDIV Setup.
• Go to "File" In that "New" to open an editor window. Create your source file and use the header file "lpc21xx.h" in the source file and save the file. Colour syntax highlighting will be
enabled once the file is saved with a extension such as ".ASM ".
• Right click on "Source Group 1" and select the option "Add Existing Files to Group
• Source Group 1"add the *.ASM source file(s) to the group. After adding the source file you can see
the file in Project Window. Then go to "Project" in that "Translate" to compile the File (s).

MODULE-2

2. **Develop and simulate ARM ALP for data Transfer, Arithmetic and Logical operations (demonstrate with the help of a suitable program)?**

```
AREA PRG6,CODE,READONLY
ENTRY;
LDR R0,=5  ;
LDR R1,=3 ;

ADD R2,R0,R1  ;
SUB R3,R0,R1  ;
MUL R4,R0,R1  ;

AND R5,R0,R1  ;
ORR R6,R0,R1  ;
EOR R7,R0,R1 ;
 END  ;
```

OUTPUT:

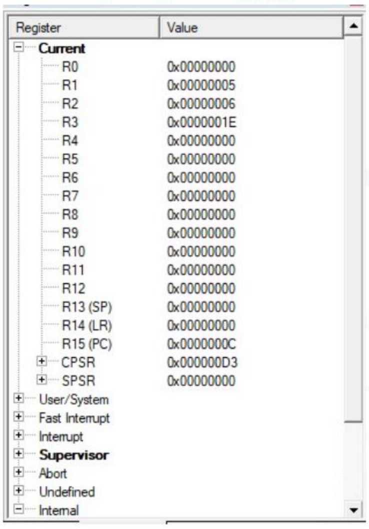


Register	Value
Current	
R0	0x00000005
R1	0x00000003
R2	0x00000008
R3	0x00000002
R4	0x0000000F
R5	0x00000001
R6	0x00000007
R7	0x00000006

3. **Develop an ALP to multiply two 16-bit binary numbers?**

```
AREA program3, CODE, READONLY
ENTRY
LDRH R1,N1;
LDRH R2,N2;
MUL R3,R1,R2;
B1 B B1;
N1 DCW 5;
N2 DCW 6;
END
```

OUTPUT:

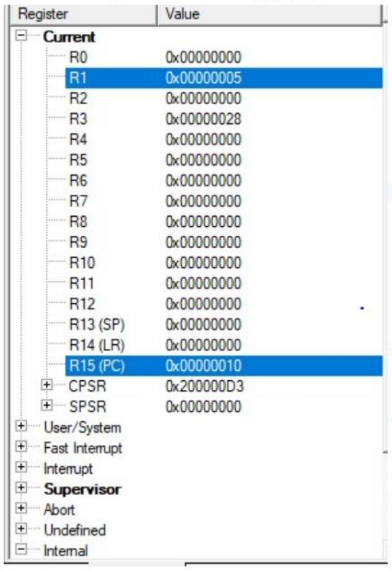


Register	Value
Current	
R0	0x00000000
R1	0x00000005
R2	0x00000006
R3	0x0000001E
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	

4. **Develop an ALP to find sum of 10 integer number?**

```
AREA program4, CODE, READONLY
ENTRY
MOV R1,#10;
MOV R2,#0;
LOOP
ADD R3,R1;
SUBS R1,#0X01;
BNE LOOP;
B1 B B1;
END
```

OUTPUT:

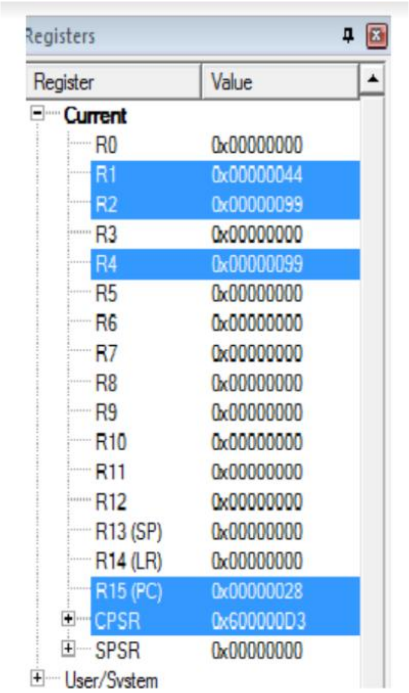


```
Register          Value
⊟ Current
    R0            0x00000000
    R1            0x00000005
    R2            0x00000000
    R3            0x00000028
    R4            0x00000000
    R5            0x00000000
    R6            0x00000000
    R7            0x00000000
    R8            0x00000000
    R9            0x00000000
    R10           0x00000000
    R11           0x00000000
    R12           0x00000000
    R13 (SP)      0x00000000
    R14 (LR)      0x00000000
    R15 (PC)      0x00000010
  ⊞ CPSR          0x200000D3
  ⊞ SPSR          0x00000000
⊞ User/System
⊞ Fast Interrupt
⊞ Interrupt
⊞ Supervisor
⊞ Abort
⊞ Undefined
⊟ Internal
```

5. **Develop an ALP to find the Largest/Smallest number in an array of 32 numbers?**

```
AREA LARGEST,CODE,READONLY
ENTRY
MOV R5,#5;
LDR R0,A
LDR R2,[R0];
NEXT ADD R0,#4;
LDR R3,[R0]
CMP R2,R3;
BHS LARGE;
MOV R2,R3;
LARGE SUBS R5,#1;
BNE NEXT;
LDR R1,RES;
STR R2,[R1];
A DCD 0X40000000;
RES DCD 0X40000020;
END
```

OUTPUT:



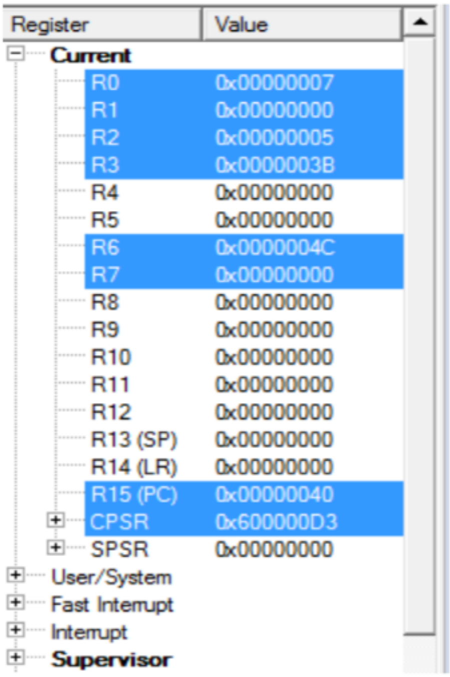
Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000044
R2	0x00000099
R3	0x00000000
R4	0x00000099
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000028
CPSR	0x600000D3
SPSR	0x00000000
User/System	

6. **Develop an ALP to count the number of ones and zeros in two consecutive memory locations?**

```
AREA Program6,CODE,READONLY
ENTRY
LDR R0,MEMORY;
LDR R1,[R0];
MOV R4,#32;
ROTATE RORS R1,#1;
BCS ONES;
ADD R3,R3,#1;
B NEXT;
ONES ADD R2,R2,#1;
NEXT ADD R4,R4,#-1;
CMP R4,#0;
BNE ROTATE;
ADD R0,R0,#04;
STRB R2,[R0];
ADD R0,R0,#1;
STRB R3,[R0];
HERE B HERE
MEMORY DCD 0X40000000;
END
```

OUTPUT:



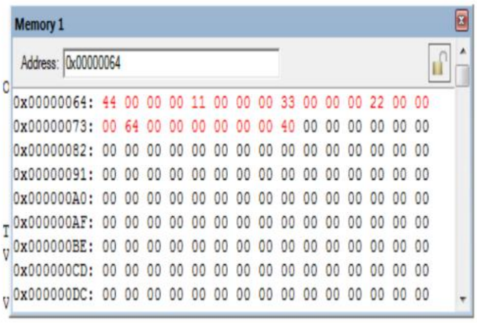
Register	Value
Current	
R0	0x00000007
R1	0x00000000
R2	0x00000005
R3	0x0000003B
R4	0x00000000
R5	0x00000000
R6	0x0000004C
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000040
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	

MODULE-3

7. Simulate a program in C for ARM microcontroller using KEIL to sort the numbers in ascending/descending order using bubble sort?

```
AREA SORT,CODE,READONLY
ENTRY
MOV R5,#5;
NXTPASS LDR R0,A;
MOV R4,R5;
NXTCOMP LDR R2,[R0];
MOV R1,R2;
ADD R0,#4;
LDR R2,[R0];
CMP R1,R2;
BLS NOEXG;
STR R1,[R0],#-4;
STR R2,[R0],#4;
NOEXG SUBS R4,#1;
BNE NXTCOMP;
SUBS R5,#1;
BNE NXTPASS;
B1 B B1
A DCD 0X40000000
END
```

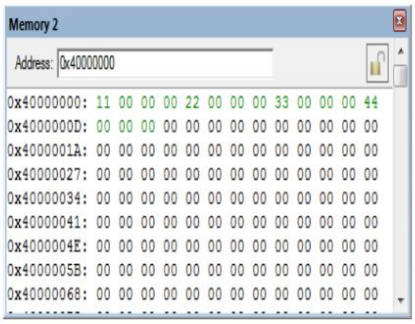
OUTPUT :



Memory 1

Address: 0x00000064

```
0x00000064: 44 00 00 00 11 00 00 00 33 00 00 00 22 00 00
0x00000073: 00 64 00 00 00 00 00 00 40 00 00 00 00 00 00
0x00000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000091: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000AF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000BE: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000CD: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000DC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



Memory 2

Address: 0x40000000

```
0x40000000: 11 00 00 00 22 00 00 00 33 00 00 00 44
0x4000000D: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000027: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000034: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000041: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000005B: 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000068: 00 00 00 00 00 00 00 00 00 00 00 00 00
```

7b)

```
#include <stdio.h>

void bubbleSort(int arr[], int n, int ascending) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if ((ascending && arr[j] > arr[j+1]) || (!ascending && arr[j] < arr[j+1])) {
                // Swap arr[j] and arr[j+1]
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int n, choice;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter the elements:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Choose sorting order:\n1. Ascending\n2. Descending\n");
    scanf("%d", &choice);
    bubbleSort(arr, n, choice == 1);

    if (choice == 1) {
        printf("Array sorted in ascending order:\n");
    } else {
        printf("Array sorted in descending order:\n");
    }

    printArray(arr, n);

    return 0;
}
```

Output

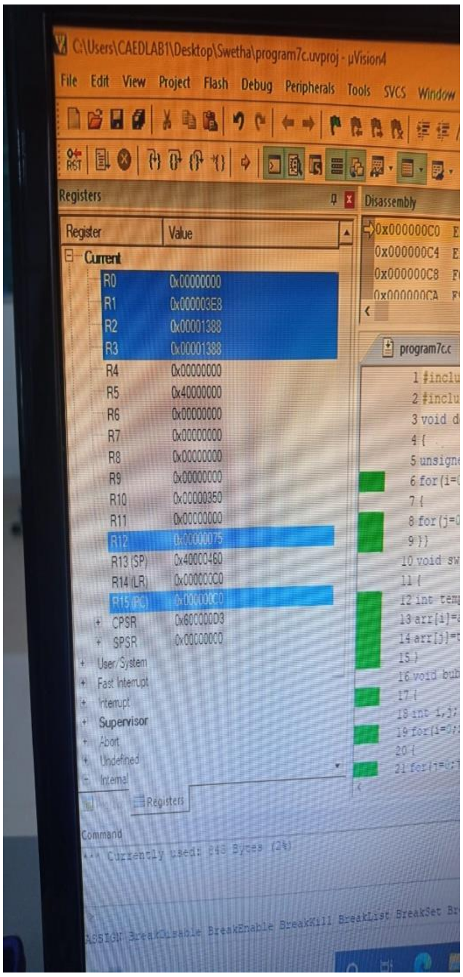
```
Enter the number of elements: 5
Enter the elements:
22
33
54
1
90
Choose sorting order:
1. Ascending
2. Descending
2
Array sorted in descending order:
90 54 33 22 1
```

7C)

```
#include<stdio.h>
#include<LPC214X.H>
#define IO0DIR  (*((volatile unsigned long*) 0xE0028008))

delay(unsigned int n)
{
       unsigned int i,j;
       for(i=0; i<n;i++)
        {
             for(j=0;j<5000;j++);
        }
}

void swap(int *arr, int i, int j)
{
       int temp = arr[i];
       arr[i] = arr[j];
       arr[j] = temp;
}

void bubblesort(int arr[ ], int n)
{
       int i,j;
                  for(i=0;i<n-1;i++)
                  {
             for(j=0;j<n-i-1;j++)
             {
                        if(arr[j]<arr[j+1])
                        {
                            swap(arr,j,j+1);
                         }
              }
                  }
}


int main( )
{
      int arr[ ] = {0x64, 0x37, 0x53,0x95, 0x21, 0x75};
      int N = sizeof(arr)/sizeof(arr[0]);
      int k;
      bubblesort(arr,N);
      for(k=0;k<6;k++)
      {
         IO0DIR = 0xffffffff;
         IO0PIN = arr[k];
         delay(1000);
      }
}
```

OUTPUT:

AFTER EXECUTION->PERIPHERALS->GPO SLOW INTERFACE->SELECT PORT 0

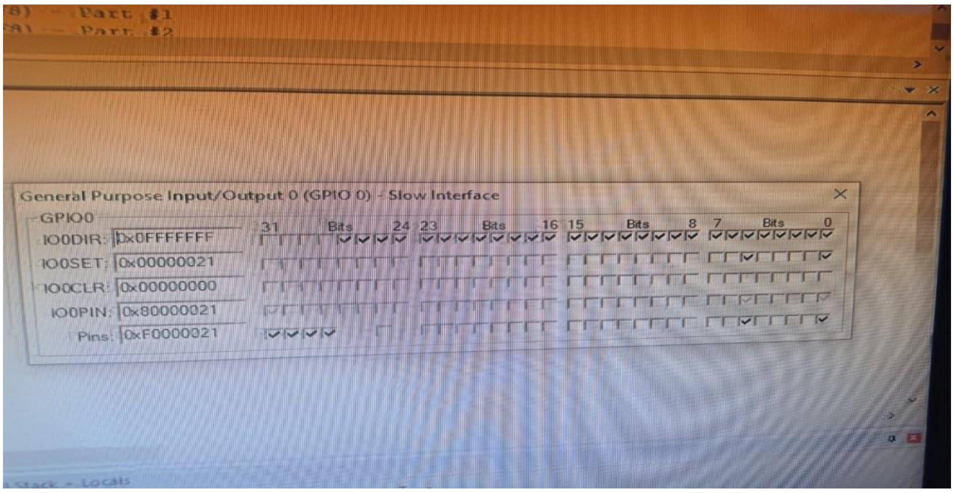


C:\Users\CAEDLAB1\Desktop\Swetha\program7c.uvproj - µVision4

```
Register    Value
Current
  R0        0x00000000
  R1        0x000003E8
  R2        0x00001388
  R3        0x00001388
  R4        0x00000000
  R5        0x40000000
  R6        0x00000000
  R7        0x00000000
  R8        0x00000000
  R9        0x00000000
  R10       0x00000350
  R11       0x00000000
  R12       0x00000075
  R13 (SP)  0x40000460
  R14 (LR)  0x000000C0
  R15 (PC)  0x000000C0
+ CPSR      0x600000D3
+ SPSR      0x00000000
```

```
0x000000C0
0x000000C4
0x000000C8
0x000000CA
```

```
*** Currently used: 848 Bytes (2%)
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet Br
```



General Purpose Input/Output 0 (GPIO 0) - Slow Interface

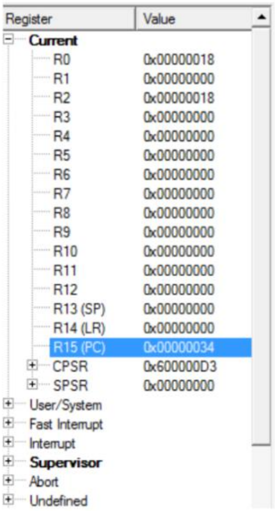
```
GPIO0
IO0DIR: 0x0FFFFFFF
IO0SET: 0x00000021
IO0CLR: 0x00000000
IO0PIN: 0x80000021
  Pins: 0xF0000021
```

8. **Simulate a program in C for ARM microcontroller to find factorial of a number?**

```
AREA FACTORIAL,CODE,READONLY
ENTRY
LDR R0,MEMORY;
LDRB R1,[R0];
MOV R2,#1;
CMP R1,#01;
BEQ STORE
MOV R2,R1;
UP ADD R1,R1,#-1
```

```
CMP R1,#0;
BEQ STORE;
MUL R3,R2,R1;
MOV R2,R3;
B UP
STORE LDR R0,RESULT
STR R2,[R0]
HERE B HERE
MEMORY DCD 0X40000000
RESULT DCD 0X40000010
END
```

OUTPUT:



Register	Value
Current	
R0	0x00000018
R1	0x00000000
R2	0x00000018
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000034
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	

b)

```
#include <stdio.h>

// Function to calculate factorial
unsigned long long factorial(int n) {
   if (n == 0 || n == 1) {
      return 1;
   }
   unsigned long long fact = 1;
   for (int i = 2; i <= n; i++) {
      fact *= i;
   }
   return fact;
}

int main() {
   int num;
   unsigned long long result;

   // Initialize UART or other communication interface here if needed
```

```
    printf("Enter a number to find its factorial: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        result = factorial(num);
        printf("The factorial of %d is %llu\n", num, result);
    }

    // Main loop if required by the microcontroller environment
    while (1) {
        // Optionally, add code to keep the microcontroller running or waiting for other tasks
    }

    return 0;
}
```

Output:

```
Enter a number to find its factorial: 5
The factorial of 5 is 120
```

9. Simulate a program in C for ARM microcontroller to demonstrate case conversion of characters from upper to lowercase and lower to uppercase?

```
#include <stdio.h>
void convertCase(char word[]) {
    int i = 0;
    while (word[i] !='\0') {
        if (word[i] >= 'A' && word[i] <='Z') {
            word[i]= word[i] -32;
        } else if (word[i] >= 'a' && word[i] <= 'z') {
            word[i] = word[i] -32;
        }
        i++;
        }
    }
int main() {
    char word[100];
    printf( "enter a word:");
    scanf("%s",word);
    printf("original word: %s\n",word);
    convertCase(word);
    printf("word after case conversion: %s\n",word);
        return 0;
}
```

Output

```
enter a word:tiger
original word: tiger
word after case conversion: TIGER
```

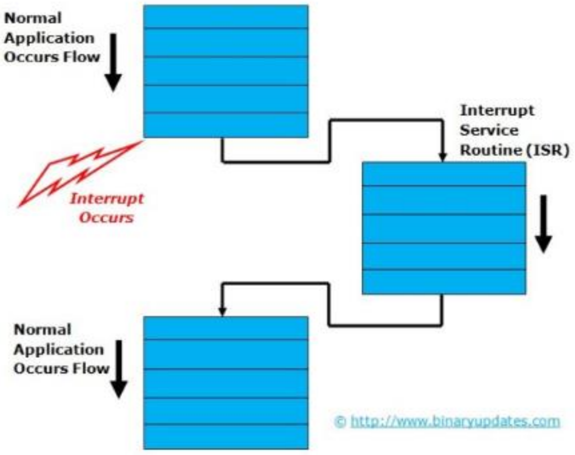
MODULE 4&5

10. Demonstrate the enabling and disabling of interrupts in ARM?

What is Interrupt or ISR?

In general, an **Interrupt** is a signal from device attached to a computer or from a program within controller that causes main program to stop and figure out what to do next. **ISR (Interrupt Service Routine)** is executed when an interrupt occurs. A section of a program that takes control when an interrupt is received and perform the operations required to service the interrupt.

How Interrupt Works?


Normal Application Occurs Flow
Interrupt Service Routine (ISR)
Interrupt Occurs
Normal Application Occurs Flow
© http://www.binaryupdates.com

Interrupt and ISR in LPC2148 ARM7 Microcontroller

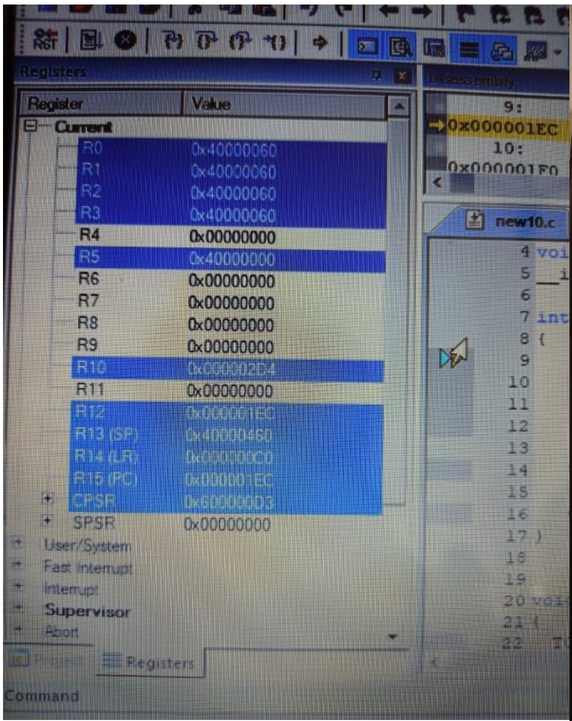
1. Whenever any device needs service of microcontroller, the device notifies the microcontroller by sending interrupt signal.
2. Upon receiving an interrupt signal, the microcontroller stops or interrupt main program flow and saves the address of the next instruction (PC) on the stack pointer (SP).
3. It jumps to a fixed location in memory, called interrupt vector table that hold the address of the ISR (Interrupt Service Routine). Each interrupt has its own ISR. The microcontroller gets the address of the ISR from the interrupt vector table and jump to it.
4. It starts to execute the Interrupt Service Routine until it reaches the last instruction of the subroutine which is RETI (Return from Interrupt). RETI not used in C Coding. (Fig: Interrupt and ISR Relation)
5. Upon executing last instruction in Interrupt Service Routine the microcontroller returns to the place where it left off or interrupted previously. And first, it gets the program counter (PC) address from the stack pointer by popping the top two bytes of the stack into the PC.
6. Then it starts to execute from that address and continue executing main program.

PROGRAM

```
  #include <lpc214x.h>
void initClocks(void);
void initTimer0(void);
 irq void timer0ISR(void);
```

```
int main(void)
        {
         initClocks();     // Initialize PLL to setup clocks
         initTimer0();     // Initialize Timer0
         IO0DIR = (1<<10);    // Configure pin P0.10 as Output
        IO0PIN = (1<<10);
         T0TCR = (1<<0);     // Enable timer
         while(1);         // Infinite Idle Loop
        }
        void initTimer0(void)
        {
         T0CTCR = 0x0;        //Set Timer Mode
        T0PR = 60000-1;     //Increment T0TC at every 60000 clock cycles
          //60000 clock cycles @60Mhz = 1 mS
          T0MR0 = 500-1;       //Zero Indexed Count-hence subtracting 1
          T0MCR = (1<<0) | (1<<1);//Set bit0 & bit1 to Interrupt & Reset TC on MR0

          VICVectAddr4 = (unsigned )timer0ISR; //Pointer Interrupt Function (ISR)
         VICVectCntl4 = (1<<5) | 4;     //(bit 5 = 1)->to enable Vectored IRQ slot
        //bit[4:0]) -> this the source number
          VICIntEnable = (1<<4);    // Enable timer0 interrupt

         T0TCR = (1<<1);         // Reset Timer
    }
        __irq void timer0ISR(void)
        {
          long int readVal;
          readVal = T0IR;     // Read current IR value
         IO0PIN ^= (1<<10);     // Toggle LED at Pin P0.10
          T0IR = readVal;     // Write back to IR to clear Interrupt Flag
          VICVectAddr = 0x0;     // End of interrupt execution
        }

        void initClocks(void)
        {
         PLL0CON = 0x01;         //Enable PLL
          PLL0CFG = 0x24;         //Multiplier and divider setup
         PLL0FEED = 0xAA;        //Feed sequence
         PLL0FEED = 0x55;

          while(!(PLL0STAT & 0x00000400)); //is locked?

          PLL0CON = 0x03;        //Connect PLL after PLL is locked
          PLL0FEED = 0xAA;       //Feed sequence
         PLL0FEED = 0x55;
         VPBDIV = 0x01;         //PCLK is same as CCLK i.e.60 MHz
        }
```

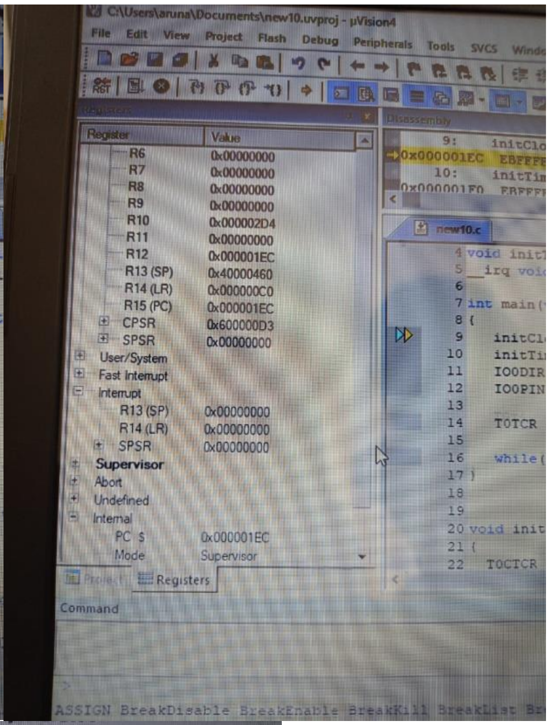
OUTPUT



```
Registers
Register        Value
Current
  R0            0x40000060
  R1            0x40000060
  R2            0x40000060
  R3            0x40000060
  R4            0x00000000
  R5            0x40000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x000002D4
  R11           0x00000000
  R12           0x000001EC
  R13 (SP)      0x40000460
  R14 (LR)      0x000000C0
  R15 (PC)      0x000001EC
  CPSR          0x600000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
```

```
Disassembly
     9:
0x000001EC
    10:
0x000001F0
```

```
new10.c
 4 voi
 5 __i
 6
 7 int
 8 {
```



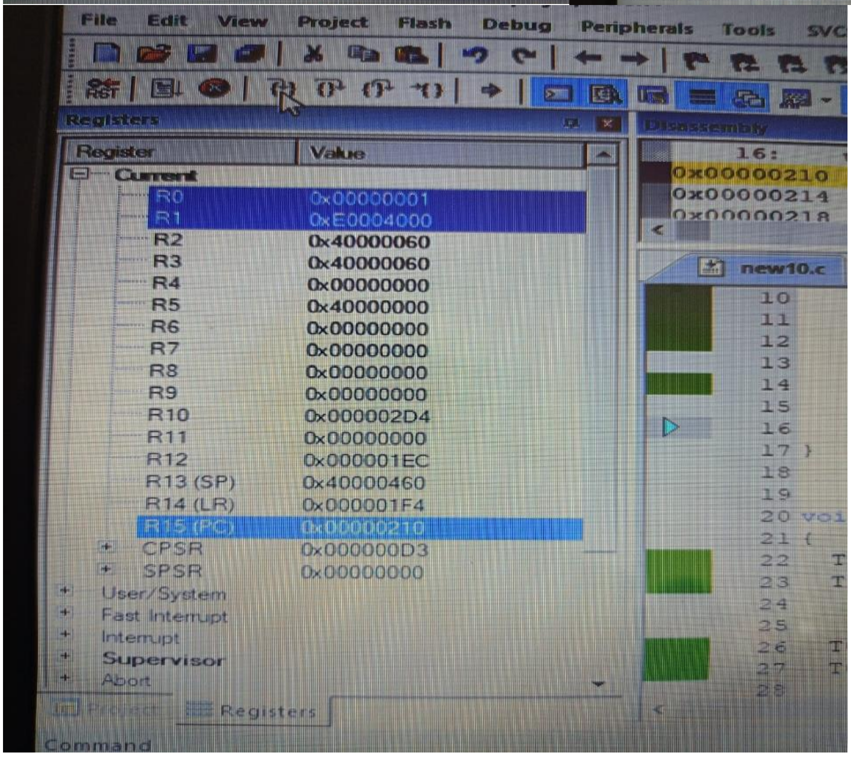
C:\Users\aruna\Documents\new10.uvproj - µVision4

```
Registers
Register        Value
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x000002D4
  R11           0x00000000
  R12           0x000001EC
  R13 (SP)      0x40000460
  R14 (LR)      0x000000C0
  R15 (PC)      0x000001EC
  CPSR          0x600000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  SPSR          0x00000000
Supervisor
Abort
Undefined
Internal
  PC $          0x000001EC
  Mode          Supervisor
```

```
Disassembly
     9:         initClo
0x000001EC  EBFFFF
    10:         initTim
0x000001F0  EBFFFF
```

```
new10.c
void initT
__irq voi

int main(
{
    initCl
    initTi
    IO0DIR
    IO0PIN

    T0TCR

    while(
}


void init
{
  T0CTCR
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList Br
```



```
Registers
Register        Value
Current
  R0            0x00000001
  R1            0xE0004000
  R2            0x40000060
  R3            0x40000060
  R4            0x00000000
  R5            0x40000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x000002D4
  R11           0x00000000
  R12           0x000001EC
  R13 (SP)      0x40000460
  R14 (LR)      0x000001F4
  R15 (PC)      0x00000210
  CPSR          0x000000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
```

```
Disassembly
    16:
0x00000210
0x00000214
0x00000218
```

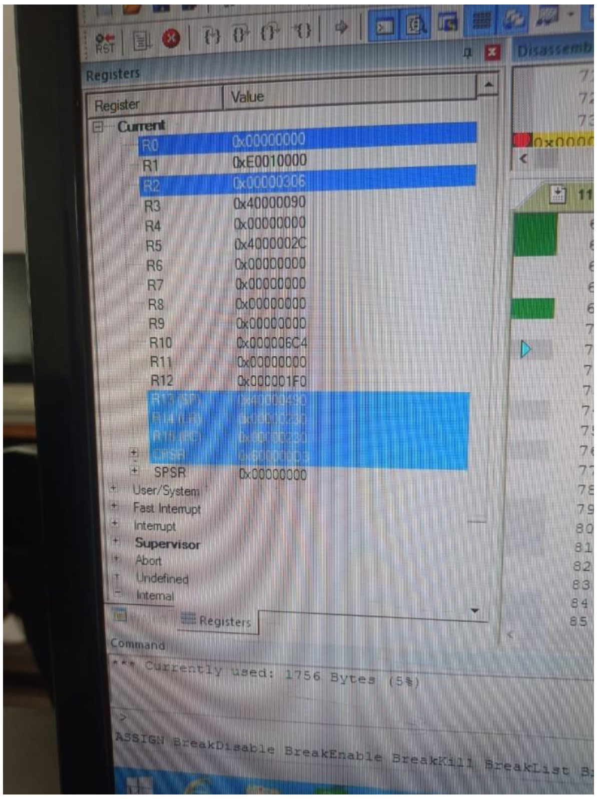
```
new10.c
10
11
12
13
14
15
16
17 }
18
19
20 voi
21 {
22    T
23    T
24
25
26    T
27    T
28
```

11. DEMONSTRATE HANDLING OF HANDLING DIVIDE BY ZERO, INVALID OPERATIONS, OVERFLOW EXCEPTION IN ARM?

```
#include <lpc214x.h>
#include <stdio.h>
#include <stdint.h>

// Function prototypes
void delay(unsigned int count);
void UART1_SendChar(char c);
void UART1_SendString(const char* string);
void UART1_IRQHandler(void);
 int result = 0;
    float inval = 0.0;
    uint32_t bigNum = UINT32_MAX;
    int successDividend = 100;
    int successDivisor = 10;
    int successDiv = 100/10;
    char successDivResult[20];

// UART initialization
void UART1_Init(void) {
    PINSEL0 |= 0x00050000;  // Select TXD1 and RXD1 for UART1
    U1LCR = 0x83;           // 8 bits, no parity, 1 stop bit, DLAB = 1
    U1DLL = 0xB7;            // 9600 baud rate for PCLK = 15MHz
    U1LCR = 0x03;           // DLAB = 0, to lock the baud rate
    U1FCR = 0x07;           // Enable and reset FIFOs
}

void delay(unsigned int count) {
    while(count--);
}

void UART1_SendChar(char c) {
    while (!(U1LSR & 0x20));  // Wait until THR is empty
    U1THR = c;                // Load the character to be transmitted
}

void UART1_SendString(const char* string) {
    while (*string) {         // Loop until the null-terminator is encountered
        UART1_SendChar(*string++);
    }
}

void UART1_IRQHandler(void) {
    volatile unsigned long IIR = U1IIR;  // Clear interrupt by reading IIR
}
```

```
int main(void) {
    UART1_Init();  // Initialize UART1
    delay(100000); // Initial delay

    // Enable UART1 interrupt
    VICVectAddr1 = (unsigned long)UART1_IRQHandler;
    VICVectCntl1 = 0x20 | 6;
    VICIntEnable = 1 << 6;

    // Declare all variables at the beginning
    //int result = 0;
    //float inval = 0.0;
    //uint32_t bigNum = UINT32_MAX;
    //int successDividend = 100;
   // int successDivisor = 10;
    //int successDiv = successDividend / successDivisor;
    //char successDivResult[20];

    // Test and send strings
    UART1_SendString("Testing UART1...\n");
    delay(1000000);  // Delay for visibility

    // Division by zero test
    UART1_SendString("Testing divide by zero...\n");
    // result = 10 / 0;  // This line will cause a runtime error
    delay(1000000);  // Delay for visibility

    // Invalid operation test
    UART1_SendString("Testing invalid operation...\n");
    // inval = inval / inval;  // This line will cause a runtime error
    delay(1000000);  // Delay for visibility

    // Overflow test
    UART1_SendString("Testing overflow...\n");
    bigNum += 1;
    delay(1000000);  // Delay for visibility

    // Successful division
    UART1_SendString("Testing successful division...\n");
    delay(1000000);  // Delay for visibility

    // Display results
    UART1_SendString("\nResults: \n");
    UART1_SendString("Divide by Zero\n");
    UART1_SendString("Invalid Operation\n");
    UART1_SendString("Overflow\n");
    UART1_SendString("Successful Division: ");
    // Send the result of successful division via UART
    sprintf(successDivResult, "%d\n", successDiv);
    UART1_SendString(successDivResult);

    while (1);
}
```

OUTPUT



```
Registers
Register        Value
Current
  R0            0x00000000
  R1            0xE0010000
  R2            0x00000306
  R3            0x40000090
  R4            0x00000000
  R5            0x4000002C
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x000006C4
  R11           0x00000000
  R12           0x000001F0
  R13 (SP)      0x40000490
  R14 (LR)      [illegible]
  R15 (PC)      [illegible]
  CPSR          0x600000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
```

```
Command
*** Currently used: 1756 Bytes (5%)

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList B
```