OUTPUT

Registers	Value
R_1	0x0000004E
R_2	0x00000063
R_3	0x00001E2A
R_{13}(SP)	0x00000000
R_{14}(LR)	0x00000004
R_{15} (PC)	0x000060F6
⊞-CPSR	0x000000D7
⊞-SPSR	0x000000D7

Date: 13/5/24
Exp. No.: 01
Expt. Title: Simple assembly language program.
Page No: 01

Using keil software, observe the various Registers Dump, CPSR, with a simple assembly language programs (ALP)

```
* MULTIPLICATION
AREA MULTIPLY, CODE, READONLY
ENTRY ; mark first instruction to execute

START
      MOV r1, #0078 ; store 1st numbers in R1
      MOV r2, #0099 ; store 2nd value in R2
MUL   MOV r3, r1, r2 ; multiplication
      NOP
      NOP
      NOP
      END                ; mark end of file.
```

OUTPUT

Registers	Value
R_0	0x000000AC
R_1	0x00000037
R_2	0x000000E3
R_{13} (SP)	0x00000000
R_{14} (LR)	0x00000004
R_{15} (PC)	0x00003194
CPSR	0x000000D7
SPSR	0x000000D7

Date: 13/5/24 | Expt. Title: Simple assembly language programs | Page No. 02

Exp. No.: 01

```
* ADDITION
    AREA ADDITION, CODE, READONLY
    ENTRY ; mark 1st instruction to execute
START
        MOV R1, #0172 ; Store 1st num in R1
        MOV R2, #0055 ; Store 2nd num in R2
        ADD R3, R1, R2 ; Addition
        NOP
        NOP
        NOP
        END                ; mark end of file
```

OUTPUT

Register	Value
R_0	0x00000000
R_1	0x000000D6
R_2	0x00000044
R_3	0x00000032
R_{13}(SP)	0x00000000
R_{14}(LR)	0x00080004
R_{15}(PC)	0x00020804
CPSR	0x000000D7
SPSR	0x000000D7

Date: 13/5/24 | Expt. Title: simple assembly language programs | Page No. 03

Exp. No.: 01

SUBTRACTION

```
    AREA SUBTRACTION, CODE, READONLY
    ENTRY ; mark 1st instruction to execute
START
    MOV  r1, #0214 ; store 1st num in R1
    MOV  r2, #0068 ; store 2nd num in R2
    SUB  r3, r1, r2 ; Subtraction.
    NOP
    NOP
    NOP
    END                ; mark end of file
```

RESULT

Observation of various registers dump, CPSR, with a simple assembly language programs (ALP) is completed.

Parameters	Max. marks	Obtained marks
Observation	4	4
Record	4	4
CIE	7	7
TOTAL	15	15

VEMANA IT

INPUT

Memory 1
Address : 0x40000000

0x40000000 : 11 21 31 41 21 22 23 24 31 32 33
0x40000000 : 51 52 00 00 00 00 00 00 00 00 00

34 41 42 43 44
00 00 00 00 00

OUTPUT

Memory 1
Address : 0x40000000

0x40000000 : 11 21 31 41 21 22 23 24 31 32 33 34 41
0x40000010 : 51 50 00 00 00 00 00 00 00 00 00 00 00
0x40000020 : 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000030 : 00 00 00 00 00 00 00 00 00 00 00 11 21
0x40000040 : 31 41 21 22 23 24 31 32 33 34 41 42 43

Date: 6/6/24
Exp. No.: 02
Expt. Title : Data transfer, logical operation and Arithmetic operations
Page No. 04

Develop and simulate ARM ALP for data transfer, arithmetic and logical operations (Demonstrate with help of progra

1. Data Transfer (BLOCK transfer from location A to location B)

```
AREA PROGRAM , CODE , READONLY
  ENTRY

START
        MOV R0 , #0X40000000; Store 1st value i
        MOV R1 , # 0X4000003C; Store 2nd value in
        MOV R4, # 0X 05 ; Store 3rd value in R4
LOOP  LDR  R2 , [R0] ; load value of R0 into R2
        STR  R2 , [R1] ; Store value of R1 into R2
        ADD R0, R0,# +4  ; Addition
        ADD R1, R1 , # +4 ; Addition
        SUBS R4 , R4 , # 0X01 ; Subtraction & u
        BNE 1001 LOOP
SAME B SAME
END
```

INPUT

Memory 1

Address : 0x40000000

0x40000000 ; 14 44 44 55 5555 55

OUTPUT

Memory 2

Address : 0x40000000

0x400000000 ; 44 44 44 44 55 55 55 55
66 6666 66

Date: 6/6/24 | Expt. Title : Data transfer, arithmetic and logical operations | Page No. 05

Exp. No.: 02

2. Arithmetic operation (Addition of two 32 bit numbers)

```
AREA,PROGRAM,CODE,READONLY
ENTRY

START
        MOV  R0 ,# 0X40000000; Store 1st value in R0
        LDR  R1 , [R0] ; load value of R0 into R1
        ADD  ,R0 , R0 , #4 ; Addition
        LDR  R2 , [R0] ; load value of R2 into R0
        ADDS R1 , R1 , R2  ; Add and update flag
        ADD  R0, R0 , #4 ; Addition
        STR  R1 , [R0] ; Store R0 value into R1.
SAME B SAME
END
```

INPUT

memory 1

Address : 0x40000000

0x40000000 : 11 01 10 11 11 11 11 11

OUTPUT

memory 1

Address : 0x40000000

0x40000000 : 11 01 10 11 11 11 11 11 11 01 10 11

Date: 6/6/24 | Expt. Title : Data transfer, arithematic and logical operation | Page No. 06

Exp. No.: 02

3. logical operation (ANDing of two 32 bit numbers)

```
    AREA Program, CODE, READONLY
    ENTRY
START
    MOV R0, #0x40000000; stor 1st value in R0
    LDR R1, [R0]; load value of R0 into R1
    ADD R0, R0, #04; Addition
    LDR R2, [R0] ; load R0 vellue into R2
    AND R1, R1, R2 ; And operation
    ADD R0, R0, #04 ; Addition
    STR R1, [R0]; store R0 value into R1.
SAME B SAME
END
```

RESULT : Simulation of ARM ALP for data transfer, arithmetic and operation is completed

Parameters	Max marks	obtained marks
Observation	4	4
Record	4	4
CIE	7	7
Total	15	15

VEMANA IT

OUTPUT

R_0	0x00000037
R_1	0x00000000
R_2	0x00000000
R_3	0x00000037
R_{14}(LR)	0x00080004
R_{15}(PC)	0x0006F670

CPSR 0x600000D7

N	0
Z	1
C	1
V	0
I	1
F	1
T	0
M	0x17

SPSR 0x600000D7

N	0
Z	1
C	1
V	0
I	1
F	1
T	0
M	0x17

Parameters	max marks	obtained marks
Record	4	4
CIE	7	4
observation	4	4
Total	15	12

Date: 6/6/24 Exp. No.: 04 Expt. Title : Sum of first 10 integers Page No. 07

4. Develop an ALP to find the sum of first 10 integer numbers

```
        AREA ADDITION, CODE, READONLY
        ENTRY
START
        MOV r0, #10; store 1st number in R0
        MOV r1, r0; stor R0 in R1
ADDIT   SUBS r1, r1, #1; subtraction & update flag
        CMP r1, #0; compeare numbers & R1
        BEQ STOP; stop if branch equal to
        ADD r3, r0, r1; Addition
        MOV r0, r3; store R3 into R0
        BNE ADDIT; if branch not equal to
STOP
        NOP
        NOP
        NOP
END
```

RESULT : Finding the sum of first 10 integer number is completed

13/6/24

OUTPUT

R_0	0x00000000
R_1	0x00000048
R_2	0x00000034
R_3	0x00000EA0
⋮	
R_4(LR)	0x00008004
R_5 (PC)	0x00042A34
CPSR -	0x00000007
SPSR -	0x00000007

Date: 13/6/24 Exp. No.: 03

Expt. Title : Multiply two 16-bit binary numbers

Page No. 08

Develop an ALP to multiply two 16-bit binary numbers

```
AREA MULTIPLY, CODE, READONLY
ENTRY ; mark first instruction to execute

START
    MOV r1, #0054 ; move 1st number to R1
    MOV r2, #0034 ; move second no to R2
    MUL r3, r1, r2 ; multiply r1 & r2 &
    NOP            store result in r3
    NOP
    END ; mark end of file.
```

RESULT : Multiplication of two 16-bit binary number is completed

13/6/24

Paramuters	max marks	obtained marks
Record	4	4
Observation	4	4
CIE	7	4
TOTAL	15	12

OUTPUT

Address : 0X40000000

```
0X40000000 : 99 99 99 99 | 88 88 88 88 | 55 55 55 55
             44 44 4444 | 22 22 22 22
0X40000017 : 22 11 11 11  11 00 00 00
             00 00 00 00  00 00 0000
             00 00 00 00
```

Current

R0	0x00000000
R1	0x4000001C
R2	0x11111111
R3	0x00000000
R4	0x4000003C
:	
:	
R15(PC)	0x00000034
CPSR	0x60000003
SPSR	0x00000000

Date: 13/6/24 Expt. Title : largest / smallest numbers

Exp. No.: 05

5> Develop an ALP to find the largest / smallest number in an array of 32 number

largest / smallest [smallest]

```
        AREA RESET, CODE, READONLY
        ENTRY ; mark 1st instruction to execute
START
        MOV R5,#6 ; store 1st number in R5
        MOV R1, #0X40000000; store second no in R
        LDR R2,[R1],#4 ; load register
LOOP
        LDR R4,[R1],#4 ; load register
        CMP R2, R4     ; compare instruction
        BHI LOOP1      ; largest smallest
        MOV R2,R4
LOOP1
        SUBS R5,R5,#1 ; Decrement counter
        CMP R5,#0 ; compare counter to 0.
        BNE LOOP
        MOV R4,#0X4000003C
        STR R2,[R4] ; stores the result in R2
        B SAME
SAME
        B SAME

        END  ; mark End of file
```

OUTPUT

ADDress : 0x40000000

```
0x40000000 : 11 11 11 11 | 22 22 22 22 |
             44 44 44 44 | 88 88 88 88

0x40000000 : 33 33 33 33 99 99 99 99
             55 55 55 55 00 00 00 00
```

current

R0	0x00000000
R1	0x40000010
R2	0x99999999
R3	0x00000000
R4	0x4000003C
.	
.	
.	
R15(PC)	0x00000034
CPSR	0x60000003
SPSR	0x00000000

Parameters	max marks	Obtained marks
Record	4	4
Observation	4	4
CIE	7	7
Total	15	15

Date: 13/6/24 Expt. Title : largest / smallest numbers. Page No. 10

Exp. No.: 05

largest number.

```
        AREA RESET, CODE, READONLY
        ENTRY ; mark 1st instruction to execute
START
        MOV R5,#6 ; store 1st number in R5
        MOV R1,#0X40000000; store 2nd no in R1
        LDR R2,[R1],#4 ; load registor
LOOP
        LDR R4,[R1],#4 ; load registor
        CMP R2,R4 ; compare instruction
        BHI LOOP1 ; largest
        MOV R2,R4
LOOP1
        ADDS R5,R5,#1 ; increment counter
        CMP R5,#0 ; compare counter to 0
        BNE LOOP
        MOV R4,#0X4000003C
        STR R2,[R4] ; stores the result in R2
        B SAME

SAME B SAME

        END ; mark end of file

RESULT ; ALP to find the largest / smallest
  number in an array of 32 number
  is completed.
```

25/6/24

Date: 13/6/24
Exp. No.: 06
Expt. Title: Count number of 1's & 0's.

write a Program to count the numbers of 1's and 0's in two consecutive memory locations.

```
    AREA SEARCH 1 , CODE , READONLY
    ENTRY ; mark 1st instruction n to execu
START
    MOV R2,#0  ; initialise Ones Counter
    MOV R3, #0 ; initialise zero Counter
    MOV R7 , #2 ; Counter to count words
    LDR R6 = DATA1 ; Loads the address of 1st
LOOP
    MOV R1 , #32 ; Counter to count no. of bits in
    LDR R0, [R6], #4 ; load the 1st value
LOOP1
    MOV R0, R0, ROR #1 ; right shift to check carr
    BHI ONE ; if carry bit is 1 jump to ONE 's
ZERO
    ADD R3, R3, #1 ; if carry bit is 0 then incre
    BSKIP ; branch to skip
ONE
    ADD R2, R2, #1 ; decrement bit counter
    BNE LOOP1 ; if not equal zero goto LOOP1
    SUBS R7, R7, #1 ; decrement word count
    CMP R7, #0 ; compare R7 to 0
    BNE LOOP ; if not equal goto LOOP
    SAME B SAME
```

OUTPUT

current

R0	0x00000001
R1	0x00000000
R2	0x00000004
R3	0x0000003C
R4	0x00000000
R5	0x00000000
R6	0x0000004C
R15(PC)	0x00000040
CPSR	0x60000D3
SPSR	0x00000000

Date: 13/6/24
Exp. No.: 06
Expt. Title: Count number of 1's & 0's
Page No. 12

```
DATA1 DCD 0x00000007, 0x00000001
END ; marks end of life
```

RESULT : Program to count the number of 1's & 0's in two consecutive memory locations completed

Particulars	max marks	obtained marks
Record	4	4
observation	4	4
CIE	7	7
Total	15	15

26/6

OUTPUT

```
Enter the number of elements : 6
Enter element 1 : 5
Enter element 2 : 1
Enter element 3 : 8
Enter element 4 : 7
Enter element 5 : 3
Enter element 6 : 0
Sorted array : 0 1 3 5 7 8
```

Date: 18/7/24 | Expt. Title : Bubble sort | Page No. 13

Exp. No.: 07

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

```
#include <lpc214x.h>
#include <stdio.h>
#include <stdlib.h>//include stdlib.h for atoi
int i;
void UART0_Init(void)
{ //UART Initialization    Enable UART0 pins P0.0[TXD0)
  PINSEL0 |= 0x00000005; // and P0.1(RXD6)
  U0LCR = 0x83; // 8-bit data, 1 stop bit, Enable DLAB
  U0DLL = 0x61; // set baud rate to 9600 (for PCLK = 15mHz)
  U0DLM = 0x00; //DLM = 0 for baud rate 9600
  U0LCR = 0x03; //Disable DLAB, 8-bit dodo, 1 stop bit
}
void UART0_SendChar(char c)
{
  while(!(U0LSR &0x20)); //wait the UART0 is ready to transmit
  U0THR = c; // Transmid character
}
char UART0_Receive char(void)
{
  while(!(U0LSR & 0x01)); //wait until data is received
  return U0RBR; //Read & return received character
}
```

Date: 28/7/24 | Expt. Title: Bubble sort | Page No. 14

Exp. No.: 07

```
void UART0_SendString(const char* Str)
{
  while(*Str)
  {
    UART0_SendChar(*Str++);
  }
}
int UART0_ReceiveInt(void)
{
  char buffer[10];
  int i=0;
  char c;
  while(1)
  {
    c=UART0_ReceiveChar();
    if(c=='\r'||c=='\n')
    {//End of input on newline or carriage return
      buffer[i]='\0';
      break;
    }
    buffer[i++]=c;
    UART0_SendChar(c);//Echo the received cha
  }
  return atoi(buffer);
}
void bubbleSort(int *arr, int n)
{
  int i, j, temp;
```

Date: 18/7/24 | Expt. Title : Bubble Sort | Page No. 15

Exp. No.: 07

```
    for (i=0; i<n-1; i++)
    {
    for (j=0; j<n-i-1; j++)
    {
     if (arr[j] > arr[j+1])
     {
       temp = arr[i];
       arr[j] = arr[j+1];
       arr[j+1] = temp;
     }
    }
   }
}
int main()
{
  int arr[20];
  int size;
  char buffer[50];

  UART0_Init();
  // Prompt & receive the number of elements
  UART0_SendString("Enter the number of
                    elements :");
  size = UART0_ReceiveInt();
  UART0_SendString("\r\n");
  // Prompt and receive array elements
  for(i=0; i<size; i++)
  {
```

Date: 18/7/24 | Expt. Title: Bubble Sort | Page No. 16

Exp. No.: 07

```
    sprintf(buffer, "Enter element %d: ", i+1);
    UART0_SendString(buffer);
    arr[i] = UART0_ReceiveInt();
    UART0_SendString("\r\n");
}
//Sort the array
bubbleSort(arr, size);
//display sorted array
UART0_SendString("Sorted array:");
for(i=0; i<size; i++)
{
  sprintf(buffer, "%d", arr[i]);
  UART0_SendString(buffer);
}
UART0_SendString("\r\n");
return 0;
}
```

Particulars	Max marks	Obtained marks
Record	4	4
Observation	4	4
CIE	7	7
Total	15	15

OUTPUT

Enter a number to calculate factorial : 5

Factorial of 5 is 120

Enter a number to calculate factorial : 15

Factorial of 15 is 2004310016

Date: 18/7/24 | Expt. Title : factorial of a number | Page No. 17

Exp. No.: 08

Simulate a Program in C for ARM microcontroller to find factorial of a number.

```
#include<lpc214x.h>
#include<stdio.h>
void UART0_Init (void)
{ // UART Initialization
  PINSEL0 |= 0x00000005; // Enable UART0 pins
      P0.0(TXD0) and P0.1 (RXD0)
  U0LCR = 0x83; //8-bit data,1 stop bit, Enable DLAB
  U0DLL = 0x61; //set baud rate to9600(for PCLK = 15m
  U0DLM = 0x00; //DLM = 0 for baud rate 9600
  U0LCR = 0x03; //Disable DLAB, 8-bit data, 1 stop b
}
void UART0_SendChar(char c)
{
  while (!(U0LSR & 0x20)); //wait until the UAR
  U0THR = c; // Transmit character
}
char UART0_ReceiveChar(void)
{
  while (!(U0LSR & 0x01)); //wait until data is re
  return U0RBR; //Read & return received charac
}
void UART0_SendString(const char *str)
{
  while(*str)
```

```
    {
      UART0_SendChar (*str++);
    }
}
int UART0_ReceiveInt(void)
{
  int number=0;
  char c;
  while (1)
  {
    c=UART0_ReceiveChar();
    if (c=='\r' || c=='\n')
    { //End of input on newline or carriage retur
      break;
    }
    if (c>='0' && c<='9')
    { //Check if the received character is a digit
      UART0_SendChar(c); //Echo the received cha
      number=number*10+(c-'0'); //Build the n
    }
  }
  return number;
}
unsigned long factorial(int n)
{
  if(n==0)
    return 1;
  else
    return n*factorial(n-1);
}
```

Date: 18/7/24 | Expt. Title: factorial of numbers | Page No. 19

Exp. No.: 08

```
int main()
{
    int num;
    unsigned long fact;
    char buffer[50];
    UART0_Init();
    // prompt and receive the number
    UART0_SendString("Enter a number to
                      calculate factorial: ");
    num = UART0_ReceiveInt();
    UART0_sendString("\r\n");
    // calculate factorial
    fact = factorial(num);
    // Display the factorial
    sprintf(buffer, "Factorial of %d is %lu
            \r\n", num, fact);
    UART0_SendString(buffer);
    return 0;
}
```

15/7/24

Particulars	max marks	obtained marks
Record	4	4
observation	4	4
CIE	7	7
Total	15	15

OUTPUT

Enter a string : DOREMON

lowercase conversion : doremon

Enter a string : doremon

uppercase conversion : DOREMON

Date: 18/7/24

Exp. No.: 09

Expt. Title : Uppercase to lowercase conversion and lowercase to uppercase conversion

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

```
#include <lpc214x.h>
//function to convert character to lowercase
char toLower(char ch);
//function to convert character to uppercase
char toUpper(char ch);

void UART0_Init(void)
{ //UART initialization
  PINSEL0 |= 0x00000005; //Enable UART0 pins P0.0
       (TXD0) & P0.1(RXD0)
  U0LCR = 0x83; //8-bit data, 1 Stop bit Enable DLAB
  U0DLL = 0x61; //Set baud rate to 9600 (for PCLK=15MHZ)
  U0DLM = 0x00; //DLM = 0 for baud rate 9600
  U0LCR = 0x03; //Disable DLAB, 8-bit data, 1 Stop bit
}
void UART0_SendChar(char c)
{
  while(!(U0LSR & 0x20)); //wait until the UART0 is
  U0THR = c; // Transmit character ready to transmit
}
```

Date: 18/07/2024
Exp. No.: 09
Expt. Title: Uppercase to lowercase conversion and lowercase to uppercase conversion


```
char UART0_ReceiveChar(void)
{
  while(!(U0LSR & 0x01)); //wait until data is received
  return U0RBR; //Read & return received character
}
Void UART0_SendString(const char *str)
{
  while(*str)
  {
    UART0_SendChar(*str++);
  }
}
Void UART0_ReceiveString(char *str, int maxLength)
{
  char c;
  int i=0;
  while(i<maxLength-1)
  {
    c=UART0_ReceiveChar();
    if(c=='\r'||c=='\n')
    {//End of input on newline or carriage return
      break;
    }
    str[i++]=c;
    UART0_SendChar(c);//Echo the received character
  }
  str[i]='\0';//Null-terminate the string
}
```

Date: 18/7/2024
Exp. No.: 09

Expt. Title : Uppercase to lowercase conversion and lowercase to uppercase conversion


```
void toLowerString(char *str)
{
  while (*str)
  {
    *str = toLower(*str);
    str++;
  }
}
void toUpperString(char *str)
{
  while (*str)
  {
    *str = toUpper(*str);
    str++;
  }
}
int main()
{
  char str[100];
  UART0_Init();
  //Prompt and convert string to lowercase
  UART0_SendString("Enter a string: ");
  UART0_ReceiveString(str, sizeof(str));
  UART0_SendString("\r\n Lowercase conversion
                                    :");

  toLowerString(str);
  UART0_SendString(str);
  UART0_SendString("\r\n");
```

```
    UART0_SendString("Enter a string:");
    UART0_ReceiveString(str, sizeof(str));
    UART0_SendString("\r\n Uppercase
                      conversion: ");
    toUpperString(str);
    UART0_SendString(str);
    UART0_SendString("\r\n");
    return 0;
}
//function definitions
char toLower(char ch) //function to convert
     character to lowercase
{
  if (ch>='A' && ch <='Z')
  {
    return ch+('a'-'A');
  }
  else
  {
    return ch; //Return unchanged if not uppercase
  }
} // function to convert character to uppercase
char toUpper(char ch)
{
  if (ch>='a' && ch <='z')
  {
    return ch-('a'-'A')
  }
```

Date: 18/07/2024
Exp. No.: 09

Expt. Title : uppercase to lowercase conversion and lowercase to uppercase conversion


```
    else
    {
        return ch; // Return unchanged if not lowercase
    }
}
```

Particulars	max marks	obtained marks
Record	4	4
observation	4	4
CIE	7	7
Total	15	15

15/7/24

OUTPUT

main 0x00000104
- a 0x00000108
- b 0x00000002
- sum 0x0000000A
- sub 0x00000006
- mul 0x00000010
- div 0x00000004

Date: 25/7/24
Exp. No.: 10
Expt. Title: C program to execute following functions & operations
Page No. 25

BEYOND SYLLABUS

Sample C program to execute addition, subtraction, multiplication & devision

```
#include<LPC21xx.h>
int main()
{
 int a=5, b=2, sum, sub, mul, div;
 sum = a+b;
 mul = a*b;
 sub = a-b;
 div = a/b;
}
```

OUTPUT

m	0x40000044C
n	0x40000438
p	0x40000424
k	0x00000005

Date: 24/7/24
Exp. No.: 11
Expt. Title: Sum of two arrays.
Page No. 26

Sum of two arrays.

```
#include<LPC21xx.h>
int main()
{
  int m[5], n[5], P[5];
  int k;
  for(k=0; k<5; k++)
  {
    P[k]=k;
    n[k]=k;
  }
  k=0;
  do
  {
    m[k] = n[k]+P[k];
     k = k+1;
  }
  while(k<5);
}
```

OUTPUT

```
i  0x00000001
x  0x00000001

i  0x00000002
x  0x00000004

i  0x00000003
x  0x00000009

i  0x00000005
x  0x00000025

i  0x00000006
x  0x00000036

i  0x00000007
x  0x00000049

i  0x00000008
x  0x00000064

i  0x00000009
x  0x00000081
```

Date: 24/4/24
Exp. No.: 12
Expt. Title: finding square from 1 to 10
Page No. 27

Find Square from 1 to 10

```
#include <LPC21xx.h>
#include <stdio.h>
int square(int i);
int main(void)
{
  int i;
  for(i=0;i<10;i++)
  {
    square(i);
  }
}
int square(int i)
{
  int x;
  x = i * i;
  return x;
}
```

24/4/24.