**Experiment-1**

**Aim: Basic operations on 32 Bit Numbers**

**PROGRAM 1**

**ADDITION OF TWO 32 BIT NUMBERS(small).**  
  
 area addition, code, readonly  
         mov r0,#0x00000009  
 mov r1,#0x00000005  
 add r2,r1,r0

l b l  
end

**PROGRAM 2**

**SUBTRACTION OF TWO 32 BIT NUMBERS(small).**  
  
 area subtraction, code, readonly  
          mov r0,#0x00000005  
mov r1,#0x00000009  
 sub r2,r1,r0

l b l  
end

**PROGRAM 3**

**MULTIPLICATION OF TWO 32 BIT NUMBERS(small).**  
  
 area multiplication, code, readonly  
          mov r0,#0x00000003  
mov r1,#0x00000002  
 mul r2,r1,r0

l b l  
end

**PROGRAM 4**

**ADDITION OF TWO 32 BIT NUMBERS(large).**  
  
 area addition, code, readonly  
          ldr r0,=0x22222222  
ldr r1,=0x44444444  
 add r2,r1,r0

l b l  
end

**PROGRAM 5**

**SUBTRACTION OF TWO 32 BIT NUMBERS(large).**  
  
 area subtraction, code, readonly  
          ldr r0,=0x33333333  
ldr r1,=0x55555555  
 sub r2,r1,r0

l b l  
end

**PROGRAM 6**

**MULTIPLICATION OF TWO 32 BIT NUMBERS(large).**  
  
 area multiplication, code, readonly  
          ldr r0,=0x12341234  
ldr r1,=0x22223333  
 umull r3,r2,r1,r0

l b l  
end

**Experiment-2**

**Aim: Basic operations on 64 Bit Numbers**

**PROGRAM 1**  
**ADDITION OF TWO 64 BIT NUMBERS.**  
        area addition, code, readonly  
              ldr r0,=0x11111111  
 ldr r1,=0x22222222  
 ldr r2,=0x33333333  
 ldr r3,=0x44444444  
 adds r4,r0,r2  
 adcs r5,r1,r3

l b l  
 end

**PROGRAM 2**  
**SUBTRACTION OF TWO 64 BIT NUMBERS.**

area subtraction, code, readonly

ldr r0,=0x55555555

ldr r1,=0x44444444

ldr r2,=0x33333333

ldr r3,=0x22222222

subs r4,r0,r2

sbc r5,r1,r3

l b l

end

**Experiment-3**

**Aim: Basic operations on Indirect addressing mode**

**PROGRAM 1: Addition of two 32 bit numbers: Data from memory**

area addind, code, readonly

ldr r0,=0x40000000

ldr r1,[r0],#04

ldr r2,[r0],#04

add r3,r2,r1

str r3,[r0]

l b l

end

**PROGRAM 2: Subtraction of two 32 bit numbers: Data from memory**

area addind, code, readonly

ldr r0,=0x40000000

ldr r1,[r0],#04

ldr r2,[r0],#04

sub r3,r2,r1

str r3,[r0]

l b l

end

**PROGRAM 3: Addition of two 64 bit numbers:**

area addind, code, readonly

ldr r0,=0x40000000

ldr r1,[r0],#04

ldr r2,[r0],#04

ldr r3,[r0],#04

ldr r4,[r0],#04

add r5,r3,r1

adc r6,r4,r2

str r5,[r0],#04

str r6,[r0]

l b l

end

**PROGRAM 4: Subtraction of two 64 bit numbers:**

area addind, code, readonly

ldr r0,=0x40000000

ldr r1,[r0],#04

ldr r2,[r0],#04

ldr r3,[r0],#04

ldr r4,[r0],#04

add r5,r3,r1

adc r6,r4,r2

str r5,[r0],#04

str r6,[r0]

l b l

end

**Experiment-4**

**Aim: Data Exchange and data transfer**

**PROGRAM 1: ADDITION OF 'N' 32 BIT NUMBERS.**  
        area nadd, code, readonly  
             ldr r0,=0x40000000  
             mov r2,#05               
             ldr r3,[r0],#04  
back      ldr r4,[r0],#04  
            add r3,r3,r4  
            subs r2,r2,#01  
            cmp r2,#00  
             bne back  
stop   b  stop  
             end

**PROGRAM 2:Data Transfer:**

area data\_trans,code,readonly

entry

ldr r0,=0x40000000

ldr r1,=0x40000044

mov r4,#09

loop ldr r2,[r0],#04

str r2,[r1],#04

subs r4,#01

cmp r4,#00

bne loop

stop b stop

end

**PROGRAM 3: Data Exchange:**

area data\_exch,code,readonly

entry

ldr r0,=0x40000000

ldr r1,=0x40000044

mov r4,#05

loop ldr r5,[r0]

ldr r6,[r1]

str r6,[r0],#04

str r5,[r1],#04

subs r4,#01

cmp r4,#00

bne loop

stop b stop

end

**Experiment-5**

**Aim: Smallest number and Largest number**

**PROGRAM 1:** WRITE AN ALP TO FIND SMALLEST NUMBER FROM THE GIVEN STRING

AREA GITAM,CODE,READONLY

ENTRY

LDR R0,=ARRAY

MOV R1,#9

LDR R2,=LS

LDR R3,[R0],#04

UP LDR R4,[R0],#04

CMP R3,R4

MOVLS R3,R4

SUB R1,R1,#01

CMP R1,#0

BNE UP

STR R3,[R2]

STOP B STOP

AREA GITAM,DATA,READONLY

ARRAY DCD 0X81,0X34,0X45,0X3,0X41,0X23,0X78,0X36,0X98,0X9

ALIGN

AREA DST,DATA

LS DCD 0

ALIGN

END

**PROGRAM 2:**we have to entr values in memory at 40000000 address and verify result at 40000060 address

area gitam,code,readonly

entry

LDR R0,=0X40000000

MOV R1,#9

LDR R2,=0X40000060

LDR R3,[R0],#04

UP LDR R4,[R0],#04

CMP R3,R4

MOVLS R3,R4

SUB R1,R1,#01

CMP R1,#0

BNE UP

STR R3,[R2]

STOP B STOP

END

**PROGRAM 3:** WRITE AN ALP TO FIND LARGEST NUMBER FROM THE GIVEN STRING

AREA GITAM,CODE,READONLY

ENTRY

LDR R0,=ARRAY

MOV R1,#9

LDR R2,=LS

LDR R3,[R0],#04

UP LDR R4,[R0],#04

CMP R3,R4

MOVHI R3,R4

SUB R1,R1,#01

CMP R1,#0

BNE UP

STR R3,[R2]

STOP B STOP

AREA GITAM,DATA,READONLY

ARRAY DCD 0X81,0X34,0X45,0X3,0X41,0X23,0X78,0X36,0X98,0X9

ALIGN

AREA DST,DATA

LS DCD 0

ALIGN

END

**PROGRAM 4:** we have to entr values in memory at 40000000 address and verify result at 40000060 address

area gitam,code,readonly

entry

LDR R0,=0X40000000

MOV R1,#9

LDR R2,=0X40000060

LDR R3,[R0],#04

UP LDR R4,[R0],#04

CMP R3,R4

MOVhi R3,R4

SUB R1,R1,#01

CMP R1,#0

BNE UP

STR R3,[R2]

STOP B STOP

END

**Experiment-6**

**Aim: Ascending and Descending Order**

**PROGRAM 1:** Arrange Ascending order (Byte data)

AREA GITAM, CODE, READONLY

ENTRY

MOV R5,#0X5

AGNN LDR R0,=0X40000000

MOV R6,#0X5

AGN LDRB R1,[R0],#01

LDRB R2,[R0]

CMP R1,R2

BLS NEXT

STRB R1,[R0]

STRB R2,[R0,#-01]

NEXT SUB R6,R6,#01

CMP R6,#00

BNE AGN

SUB R5,R5,#01

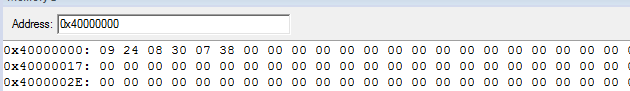
CMP R5,#00

BNE AGNN

L B L

END

I/P: 9,36,8,48,7,56



O/P: 7,8,9,36,48,56 Graphical user interface, text, application

Description automatically generated

**PROGRAM 2:** Arrange Descending order (Byte data).

AREA GITAM, CODE, READONLY

ENTRY

MOV R5,#0X5

AGNN LDR R0,=0X40000000

MOV R6,#0X5

AGN LDRB R1,[R0],#01

LDRB R2,[R0]

CMP R1,R2

BHI NEXT

STRB R1,[R0]

STRB R2,[R0,#-01]

NEXT SUB R6,R6,#01

CMP R6,#00

BNE AGN

SUB R5,R5,#01

CMP R5,#00

BNE AGNN

L B L

END

I/P: 9,36,8,48,7,56

Graphical user interface, text, application

Description automatically generated

O/P: 56,48,36,9,8,7Graphical user interface, text, application

Description automatically generated

**PROGRAM 3:** Arrange Ascending order (Halfword data)

AREA GITAM, CODE, READONLY

AREA GITAM, CODE, READONLY

ENTRY

MOV R5,#0X5

AGNN LDR R0,=0X40000000

MOV R6,#0X5

AGN LDRH R1,[R0],#02

LDRH R2,[R0]

CMP R1,R2

BLS NEXT

STRH R1,[R0]

STRH R2,[R0,#-02]

NEXT SUB R6,R6,#01

CMP R6,#00

BNE AGN

SUB R5,R5,#01

CMP R5,#00

BNE AGNN

L B L

END

Data : 84,59,36,78,39,47,85,91,27,64,56,23

I/P: 3B54, 4E24, 2F27, 5B55, 401B, 1738

Graphical user interface, application

Description automatically generated

O/P: 1738, 2F27, 3B54, 401B, 4E24, 5B55

Graphical user interface, text

Description automatically generated

**PROGRAM 4:** Arrange Descending order (Halfword data).

AREA GITAM, CODE, READONLY

ENTRY

MOV R5,#0X5

AGNN LDR R0,=0X40000000

MOV R6,#0X5

AGN LDRH R1,[R0],#02

LDRH R2,[R0]

CMP R1,R2

BHI NEXT

STRH R1,[R0]

STRH R2,[R0,#-02]

NEXT SUB R6,R6,#01

CMP R6,#00

BNE AGN

SUB R5,R5,#01

CMP R5,#00

BNE AGNN

L B L

END

Data : 84,59,36,78,39,47,85,91,27,64,56,23

I/P: 3B54, 4E24, 2F27, 5B55, 401B, 1738

Graphical user interface, text, application

Description automatically generated

O/P: 5B55, 4E24, 401B, 3B54, 2F27, 1738

Graphical user interface, text, application

Description automatically generated

**PROGRAM 5:** Arrange Ascending order (word data)

AREA GITAM, CODE, READONLY

ENTRY

MOV R5,#0X5

AGNN LDR R0,=0X40000000

MOV R6,#0X5

AGN LDR R1,[R0],#04

LDR R2,[R0]

CMP R1,R2

BLS NEXT

STR R1,[R0]

STR R2,[R0,#-04]

NEXT SUB R6,R6,#01

CMP R6,#00

BNE AGN

SUB R5,R5,#01

CMP R5,#00

BNE AGNN

L B L

END

DATA : 92,83,47,36,47,91,26,72,45,88,99,34,64,58,23,56,76,87,90,06,87,02,56,57

I/P: 242F535C, 481A5B2F, 2263582D, 38173A40, 065A574C, 39380257

Graphical user interface, text, application

Description automatically generated

O/P: 065A574C, 2263582D, 242F535C, 38173A40, 39380257, 481A5B2F

Graphical user interface, text, application

Description automatically generated

**PROGRAM 6:** Arrange Descending order (word data).

AREA GITAM, CODE, READONLY

ENTRY

MOV R5,#0X5

AGNN LDR R0,=0X40000000

MOV R6,#0X5

AGN LDR R1,[R0],#04

LDR R2,[R0]

CMP R1,R2

BHI NEXT

STR R1,[R0]

STR R2,[R0,#-04]

NEXT SUB R6,R6,#01

CMP R6,#00

BNE AGN

SUB R5,R5,#01

CMP R5,#00

BNE AGNN

L B L

END

DATA: 92,83,47,36,47,91,26,72,45,88,99,34,64,58,23,56,76,87,90,06,87,02,56,57

I/P: 242F535C, 481A5B2F, 2263582D, 38173A40, 065A574C, 39380257

Graphical user interface, text, application

Description automatically generated

O/P: 481A5B2F, 39380257, 38173A40, 242F535F, 2263582D, 065A574C

Graphical user interface, text, application

Description automatically generated

**Experiment-7**

**Aim: Finding No of 1’s and 0’s in given number and verifying given number is even or odd**

**PROGRAM 1: Count No of 1’s and 0’s in given Number**

AREA GITAM, CODE, READONLY

ENTRY

MOV R2,#0

MOV R3,#0

LDR R0,=0X40000000

MOV R1,#32

LDR R5,[R0]

LOOP2 MOVS R5,R5,ROR #1

BHI ONES

ADD R3,R3,#1

B LOOP1

ONES ADD R2,R2,#1

LOOP1 SUB R1,R1,#1

CMP R1,#00

BNE LOOP2

L B L

END

**PROGRAM 2. Separate even and odd numbers**

AREA GITAM, CODE, READONLY

ENTRY

MOV R2,#0

MOV R3,#0

MOV R1,#8

LDR R0,=0X40000000

LDR R7,=0X40000030

LDR R8,=0X40000050

LOOP2 LDR R5,[R0],#1

MOV R6,R5

MOVS R5,R5,ROR #1

BHI ODD

ADDS R3,R3,#1

STR R6,[R8],#1

B LOOP1

ODD ADD R2,R2,#1

STR R6,[R7],#1

LOOP1 SUB R1,R1,#1

CMP R1,#00

BNE LOOP2

L B L

END

**Experiment-7**

**Aim: Blinking of LED and Generation of Waveforms**

**PROGRAM 1: Write an ALP to Blinking LED Waveform**

AREA GITAM, CODE, READONLY

ENTRY

IO0DIR EQU 0XE0028008

IO0SET EQU 0XE0028004

IO0CLR EQU 0XE002800C

IO0PIN EQU 0XE0028000

PINSEL0 EQU 0xE002C000

LDR R0,=PINSEL0

LDR R1,=0X00000200

STR R1,[R0]

LDR R0,=IO0DIR

LDR R2,=0X00000200

STR R2,[R0]

NEXT LDR R2,=0X00000200

LDR R0,=IO0SET

STR R2,[R0]

BL DELAY

LDR R0,=IO0CLR

STR R2,[R0]

BL DELAY

B NEXT

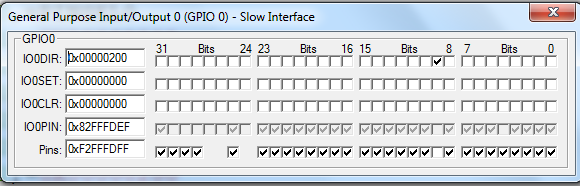
DELAY LDR R5,=0X0000FFFF

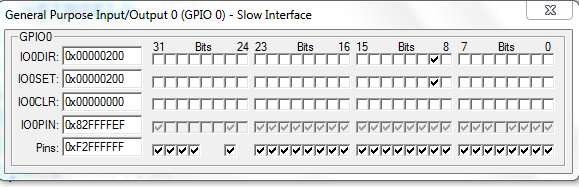
AGN SUBS R5,R5,#0X01

BNE AGN

BX LR

END

**OUTPUT **

****

**PROGRAM 2: Write an ALP to Generate Square waveform**

AREA GITAM, CODE, READONLY

ENTRY

PINSEL1 EQU 0XE002C004

DACR EQU 0XE006C000

LDR R1,=PINSEL1

LDR R2,=0X00080000

STR R2,[R1]

LDR R0,=DACR

LDR R2,=0X00000000

LDR R3,=0X000003FF

BAK LSL R1,R3,#6

STR R1,[R0]

BL DELAY

LSL R1,R2,#6

STR R1,[R0]

BL DELAY

B BAK

DELAY LDR R5,=0X0000FFFF

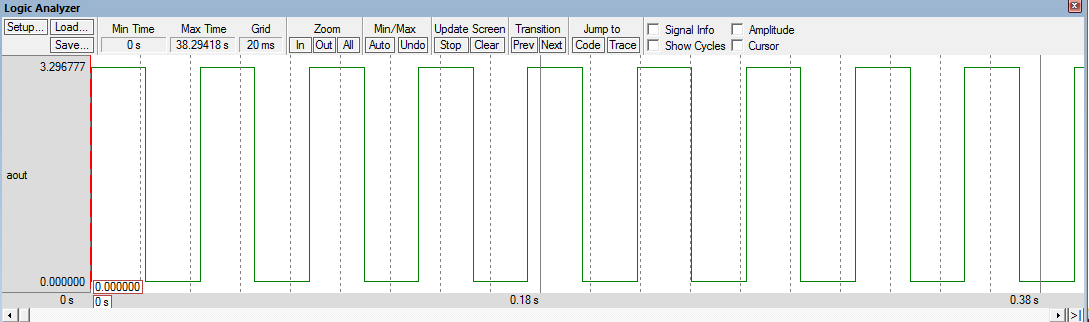
AGN SUBS R5,R5,#0X01

BNE AGN

BX LR

END

**OUTPUT**

****

**PROGRAM 3: Write an ALP to Generate Sawtooth Waveform**

AREA GITAM, CODE, READONLY

ENTRY

PINSEL1 EQU 0XE002C004

DACR EQU 0XE006C000

LDR R3,=0X0000003FF

LDR R1,=PINSEL1

LDR R2,=0X00080000

STR R2,[R1]

LDR R0,=DACR

BAK1 LDR R2,=0X00

BAK LSL R1,R2,#6

STR R1,[R0]

ADD R2,R2,#1

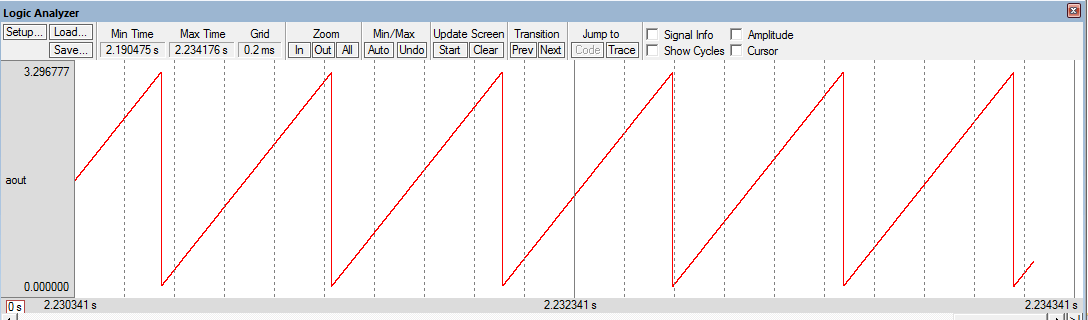
CMP R2,R3

BLT BAK

B BAK1

END

**OUTPUT WAVEFORM**



**PROGRAM 4: Write an ALP to Generate Triangular Waveform**

AREA GITAM, CODE, READONLY

ENTRY

PINSEL1 EQU 0XE002C004

DACR EQU 0XE006C000

LDR R3,=0X0000003FF

LDR R1,=PINSEL1

LDR R2,=0X00080000

STR R2,[R1]

LDR R0,=DACR

LDR R2,=0X00

BAK LSL R1,R2,#6

STR R1,[R0]

ADD R2,R2,#1

CMP R2,R3

BLT BAK

BAK1 LSL R1,R2,#6

STR R1,[R0]

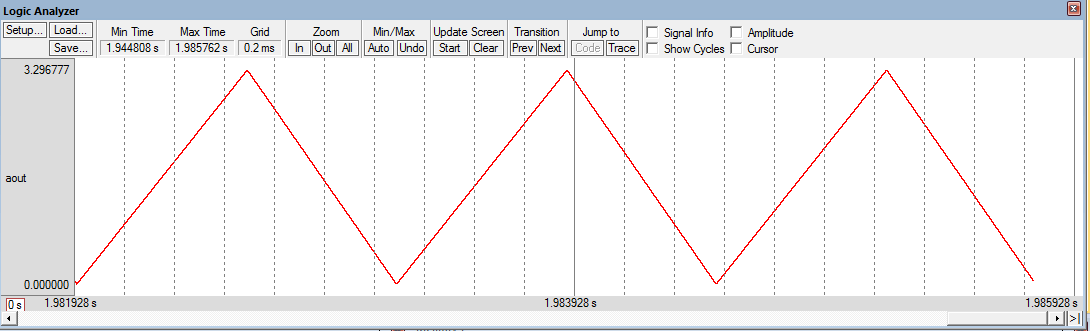
SUBS R2,R2,#1

BNE BAK1

B BAK

END

**OUTPUT WAVEFORM**



**Experiment-8**

**Aim: Blinking of LED**

#include<lpc214x.h>

void delay();

void main()

{

IO0DIR |=0XfffffFFF; //Port 0 is now acting as a output pin

while(1) {

IOSET0 |=0XfffffFFF; //Port 0's all pins are high now (LED is glowing)

delay();

IOCLR0 |=0XFFFfffff; //Port 0's all pins are low now (LED is OFF)

delay();

}

}

void delay()

{

unsigned int i;

for(i=0;i<30000;i++);

}

**Experiment-9**

**Aim: Generation of delay using Times**

#include<lpc214x.h>

void delay(unsigned int z);

void pll();

int main(void)

{

IO0DIR=0xffffffff;

pll(); //Fosc=12Mhz,CCLK=60Mhz,PCLK=60MHz

while(1) {

IO0SET=0xffffffff;

delay(1000); //1sec delay

IO0CLR=0xffffffff;

delay(1000); //1sec delay

}

}

void pll() //Fosc=12Mhz,CCLK=60Mhz,PCLK=60MHz

{

PLL0CON=0x01;

PLL0CFG=0x24;

PLL0FEED=0xaa;

PLL0FEED=0x55;

while(!(PLL0STAT&(1<<10)));

PLL0CON=0x03;

PLL0FEED=0xaa;

PLL0FEED=0x55;

VPBDIV=0x01;

}

void delay(unsigned int z)

{

T0CTCR=0x0; //Select Timer Mode

T0TCR=0x00; //Timer off

T0PR=59999; //Prescaler value for 1ms

T0TCR=0x02; //Timer reset

T0TCR=0x01; //Timer ON

while(T0TC<z);

T0TCR=0x00; //Timer OFF

T0TC=0; //Clear the TC value. This is Optional.

}