**Mohsen Zarei (s343388)**

**ASSAMBLY SECTION**

100% correct. based on given values it returns appropriate results. (y = 25, n = 4 🡪 RESULT = 0xFFFFFFB0 | y = -24, n = 6 🡪RESULT = 0xFFFFFFB6 | y = 27, n = 4 🡪 RESULT = 0xFFFFFFA6)

!!!Attention!!! For testing the ASM codes solely, SystemInit(); need to be commented on in Main.c!!!

**- startup\_LPC17xx.s:**

….

AREA |.text|, CODE, READONLY

; Reset Handler

Reset\_Handler PROC

EXPORT Reset\_Handler [WEAK]

IMPORT \_\_main

LDR R0, =\_\_main

BX R0

;;;;;;for assembly test SystemInit(); in main.c need to comment

IMPORT Maclaurin\_cos

LDR R0,=25 ;Y

LDR R1,=4 ;N

BL Maclaurin\_cos

stop B stop

ENDP

…..

**- Maclaurin\_cos.s:**

;LDR R0,=25 ;Y

;LDR R1,=4 ;N

;\*\*\*\*\*\*\*\*\* ;for result test only \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; AREA TEST, DATA, READWRITE

;SPA SPACE 8

;\*\*\*\*\*\*\*\*\* ;for result test only \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AREA exam, CODE, READONLY

Maclaurin\_cos PROC

PUSH{R4-R10,LR} ; {R4-R8, R10, R11, PC}

EXPORT Maclaurin\_cos

I RN 3

ITR RN 6

;LDR R4,=SPA ;for result test only

MOV R9,#0

LDR I,=100

MOV R10,I

LDR ITR,=1 ;number of iterations

FOR

MOV R5,#0

SUBS R5,I ;making negative number -t(i-1)

MUL R2,R0,R0 ;R2=y^2

MUL R5,R2 ;R5 = -t(i-1)· y^2

;STR R5,[R4]

MOV R7,#2

MUL R7,ITR ;2i

SUBS R8,R7,#1

MUL R8,R7

MOV R7,#100

MUL R8,R7 ;R8 = (2i-1)·(2i)· 100

;STR R8,[R4] ;for result test only

SDIV I,R5,R8

ADD R9,I

;STR I,[R4],#4 ;for result test only

ADD ITR,#1

CMP ITR,R1

BLS FOR

ADD R0,R9,R10

POP{R4-R10,PC} ; {R4-R8, R10, R11, PC}

ENDP

END

**ARM-C SECTION**

**- Main.c:**

iint main(){

SystemInit(); //need to comment for ASMBELY testing

BUTTON\_init();

//DAC pin configuration:

LPC\_PINCON->PINSEL1 |= (1<<21);

LPC\_PINCON->PINSEL1 &= ~(1<<20);

LPC\_GPIO0->FIODIR |= (1<<26);

while(1){

}

}

**- lib\_button.c:**

#include "button.h"

#include "lpc17xx.h"

void BUTTON\_init(void) {

LPC\_PINCON->PINSEL4 |= (1 << 20); /\* External interrupt 0 pin selection \*/

LPC\_GPIO2->FIODIR &= ~(1 << 10); /\* PORT2.10 defined as input \*/

LPC\_PINCON->PINSEL4 |= (1 << 22); /\* External interrupt 0 pin selection \*/

LPC\_GPIO2->FIODIR &= ~(1 << 11); /\* PORT2.11 defined as input \*/

LPC\_PINCON->PINSEL4 |= (1 << 24); /\* External interrupt 0 pin selection \*/

LPC\_GPIO2->FIODIR &= ~(1 << 12); /\* PORT2.12 defined as input \*/

LPC\_SC->EXTMODE = 0x7;

NVIC\_EnableIRQ(EINT2\_IRQn); /\* enable irq in nvic \*/

NVIC\_EnableIRQ(EINT1\_IRQn); /\* enable irq in nvic \*/

NVIC\_EnableIRQ(EINT0\_IRQn); /\* enable irq in nvic \*/

}

**- IRQ\_button.c:**

#include "LPC17xx.h"

#include "../Main.h"

#include "LPC17xx.h"

void EINT1\_IRQHandler (void)

{

init\_timer\_SRI(1,1592,0b011); //stop reset interrupt

enable\_timer(1);

// LPC\_SC->EXTINT &= (1 << 1); /\* clear pending interrupt \*/

}

**- lib\_timer.c:**

#include "LPC17xx.h"

#include "timer.h"

uint32\_t tick=0;

void enable\_timer( uint8\_t timer\_num )

{

if ( timer\_num == 0 )LPC\_TIM0->TCR = 1;

else if ( timer\_num == 1 )LPC\_TIM1->TCR = 1;

else if ( timer\_num == 2 )LPC\_TIM2->TCR = 1;

else if ( timer\_num == 3 )LPC\_TIM3->TCR = 1;

return;

}

uint32\_t read\_timer( uint8\_t timer\_num )

{

if ( timer\_num == 0 ) return LPC\_TIM0->TC;

else if ( timer\_num == 1 ) return LPC\_TIM1->TC;

else if ( timer\_num == 2 ) return LPC\_TIM2->TC;

else if ( timer\_num == 3 ) return LPC\_TIM3->TC;

return 0;

}

void disable\_timer( uint8\_t timer\_num )

{

if ( timer\_num == 0 ) LPC\_TIM0->TCR = 0;

else if ( timer\_num == 1 ) LPC\_TIM1->TCR = 0;

else if ( timer\_num == 2 ) LPC\_TIM2->TCR = 0;

else if ( timer\_num == 3 ) LPC\_TIM3->TCR = 0;

return;

}

void reset\_timer( uint8\_t timer\_num )

{ uint32\_t regVal;

if( timer\_num == 0 ){

regVal = LPC\_TIM0->TCR;

regVal |= 0x02;

LPC\_TIM0->TCR = regVal;

}else if ( timer\_num == 1 ){

regVal = LPC\_TIM1->TCR;

regVal |= 0x02;

LPC\_TIM1->TCR = regVal;

}return;

}

uint32\_t init\_timer\_SRI ( uint8\_t timer\_num, uint32\_t TimerInterval, uint32\_t configuration )

{

if ( timer\_num == 0 )

{

LPC\_TIM0->MR0 = TimerInterval;

LPC\_TIM0->MCR = configuration;

NVIC\_EnableIRQ(TIMER0\_IRQn);

/\*NVIC\_SetPriority(TIMER0\_IRQn, 4);\*/ /\* less priority than buttons \*/ //priorities need to not be the same

//NVIC\_SetPriority(TIMER0\_IRQn, 0); /\* more priority than buttons \*/

return (1);

}else if ( timer\_num == 1 )

{ LPC\_TIM1->MR0 = TimerInterval;

LPC\_TIM1->MCR = configuration;

NVIC\_EnableIRQ(TIMER1\_IRQn);

//NVIC\_SetPriority(TIMER1\_IRQn, 5); /\* less priority than buttons and timer0\*/

return (1);

} }

return (0);

}}

**- IRQ\_timer.c:**

#include "LPC17xx.h"

#include "../Main.h"

int cosineValues[45] = {0};

extern int Maclaurin\_cos(int input1,int input2);

void TIMER1\_IRQHandler (void)

{

static int repeat = 0;

static int ticks = 0;

int input, output;

if (repeat < 200)

{

float tmp input = (1.428 \* ticks); // see Note 1 below

if(tmp input>=0){

tmp input+=0.5;

}

else if(tmp input<0){

tmp input-=0.5;

}

input = (int)tmp; //casting a float value to an integer

output = 500 + Maclaurin\_cos(input, 3) / 2;

cosineValues[ticks + 22] = output; // see Note 2 below

DAC\_write(output);

ticks ++;

if (ticks > 22)

{

ticks = -22;

repeat += 1;

}

else DAC\_write(0);

}

LPC\_TIM1->IR = 1; /\* clear interrupt flag \*/

return;

}

**- DAC.c:**

#include "LPC17xx.h"

void DAC\_write(uint16\_t value){

LPC\_DAC->DACR = value<<6;

}