**Mohsen Zarei (s343388)**

Kind professor, due to the difficulties of the questions and the limited time, I couldn’t implement all my solutions in the appropriate way. Anyway, after revision, I tested the project on the board, and it works well.

Having said that, and since I attended the **PacMan** project and **all** **laboratories**, I hope my revision can be accepted. Thank you in advance for your consideration. I'm available for any further information.

Revision guideline:

Unwritten sections by: “…..”

removed sections: Highlighted in red

Modified/added sections: Highlighted in Yellow

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

**ASSAMBLY SECTION**

**- startup\_LPC17xx.s:**

….

AREA DD, DATA, READONLY

MAT\_A DCB 0xF8, 0x7C, 0x3E, 0x1F, 0x8F, 0xC7, 0xE3, 0xF1

AREA DDD,DATA, READWRITE

MAT\_AT SPACE 8

AREA |.text|, CODE, READONLY

; Reset Handler

Reset\_Handler PROC

EXPORT Reset\_Handler [WEAK]

IMPORT \_\_main

LDR R0, =\_\_main

BX R0

;1. the address of the bit matrix A

;2. the address of the bit matrix AT

IMPORT transposition

LDR R0,=MAT\_A

LDR R1,=MAT\_AT

BL transposition

stop B stop

ENDP

…..

**- Transposition subroutine:**

AREA |.text|, CODE, READONLY

transposition PROC

EXPORT transposition

PUSH {R4-R8,LR} ;I wanted to check finally, but time was limit!

I RN 2

J RN 3

MOV I,#0

MOV R7,#7

FORI ;for (i = 0; i < 8; i ++)

CMP I, #8

BHS ENDFORI

LDRB R4,[R0,I] ;x = i-th row of A

LDR J, =70

FORJ ;for (j = 0; j < 8; j ++)

CMP J, #08 ;loop logic reversed

BEQHS ENDFORJ

SUB R8,R7,J ;pointer for shifting MSB to LSB position

LSR R5,R4, J R8

AND R5,#1

SUB R8,R7,I

LSL R5, J R8

LDRB R6,[R1,J]

ORR R6,R5 ;b = j-th bit of x

STRB R5R6,[R1,J] ;store b to the i-th bit of the j-th row of AT

;LSR R5,R4,J ;unnecessary duplications!

;AND R5,R4,J

SUB ADD J, #1 ;loop logic reversed B FORJ

ENDFORJ

ADD I, #1

B FORI

ENDFORI

;result

POP {R4-R8,PC}

ENDP

END

**ARM-C SECTION**

**- Main:**

int main(){

SystemInit();

LPC\_SC->PCONP |= (1 << 22); //power control timer 2 enabled from wizard

init\_timer\_SRI(2,0xFFFF,0b010);

enable\_timer(2);

BUTTON\_init();

LED\_init();

init\_RIT(0x004C4B40); //due to need for debouncing keys!

enable\_RIT();

while(1){

}

}

**- IRQ Button:**

Since in the question didn’t mentioned to debouncing, codes in this section are totally shifted to RIT-Handler, while this implementation could be easily done during the exam!! Hope for more mercy.

#include "../Main.h"

#include "LPC17xx.h"

char A[8];

char B[8];

char array[];

int indexa=0;

int indexb=0;

char transposition(char \*a, char \*b);

extern int KEY0;

extern int KEY1;

extern int KEY2;

void EINT0\_IRQHandler (void)

{

for (int i=0;i<3;i++){

array[i] = transposition(A, B);}

if (array[1]+array[2] == (A[8]+B[8])){

LED\_On(1);

}else LED\_On(2);

NVIC\_DisableIRQ(EINT0\_IRQn); /\* disable Button interrupts \*/

LPC\_PINCON->PINSEL4 &= ~(1 << 20); /\* GPIO pin selection \*/

KEY0=1;

LPC\_SC->EXTINT &= (1 << 0);

}

void EINT1\_IRQHandler (void)

{

A[indexa] = (char)read\_timer(2);

indexa++;

NVIC\_DisableIRQ(EINT1\_IRQn);

LPC\_PINCON->PINSEL4 &= ~(1 << 22);

KEY1=1;

LPC\_SC->EXTINT &= (1 << 1);

}

void EINT2\_IRQHandler (void)

{

B[indexb] = (char)read\_timer(2);

indexb++;

NVIC\_DisableIRQ(EINT2\_IRQn);

LPC\_PINCON->PINSEL4 &= ~(1 << 24);

KEY2=1;

LPC\_SC->EXTINT &= (1 << 2);

}

**- IRQ RIT:**

Didn’t highlighted untouched sections came from Button IRQ

#include "../Main.h"

#include "LPC17xx.h"

char A[8], B[8];

char AT[8], BT[8], SUM[8], SUMT[8];

int indexa=0, indexb=0, result = 1;

volatile int KEY0=0, KEY1=0, KEY2=0;

extern void transposition(char \*Tabel, char \*Transposed);

void RIT\_IRQHandler (void){

if (KEY0 >= 1) { if ((LPC\_GPIO2->FIOPIN & (1 << 10)) == 0) { // KEY0 Pressed

switch (KEY0) {

case 2: // code section KEY0

transposition(A, AT);

transposition(B, BT);

for (int i = 0; i < 8; i++) SUM[i] = A[i] ^ B[i];

transposition(SUM, SUMT);

for (int i = 0; i < 8; i++) {

if (SUMT[i] != (AT[i] ^ BT[i])) {

result = 0;

break; }}

if (result == 1) LED\_On(1);

else LED\_On(2);

break;

default:

break;}

KEY0++;

} else { // Button released

KEY0 = 0;

NVIC\_EnableIRQ(EINT0\_IRQn); // Enable Button interrupts

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

}}

if (KEY1 >= 1) { if ((LPC\_GPIO2->FIOPIN & (1 << 11)) == 0) { // KEY1 Pressed

switch (KEY1) {

case 2: // code section KEY1

if (indexa < 8) {A[indexa] = (char)read\_timer(2); indexa++;}

break;

default:

break;}

KEY1++;

} else {

KEY1 = 0;

NVIC\_EnableIRQ(EINT1\_IRQn);

LPC\_PINCON->PINSEL4 |= (1 << 22);

}}

if (KEY2 >= 1) { if ((LPC\_GPIO2->FIOPIN & (1 << 12)) == 0) { // KEY2 Pressed

switch (KEY2) {

case 2: // code section KEY2

if (indexb < 8) {B[indexb] = (char)read\_timer(2); indexa++;}

break;

default:

break;}

KEY2++;

} else {

KEY2 = 0;

NVIC\_EnableIRQ(EINT2\_IRQn);

LPC\_PINCON->PINSEL4 |= (1 << 24);

}}

reset\_RIT();

LPC\_RIT->RICTRL |= 0x1;

return;

}