@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

@ ECE 372 – PROGRAMMING PROJECT I

@--------------------------------------------------------------------------------

@ USER Rotate\_LEDS, TIMERS INTERRUPT, AND BUTTON INTERRUPT

@ This program will rotating four USR Rotate\_LEDS when the button is pressed

@ The USER Rotate\_LEDS will rotate from USER\_LED 3 to USER LED\_0 and go back to

@ USER\_LED 3. Timer generates a delay 1 second compared with real life.

@-------------------------------------------------------------------------------

@ Hai Dang Hoang

@ Reference: Douglas V. Hall and Leela Yadlapalli

@ January 18th, 2015

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

**.text**

**.global** \_start

**.global** INT\_DIRECTOR

**\_start:**

@INITIALIZE STACK-----------------------------------------------------------------@

LDR R13,=STACK1 @ Point to base of STACK for SVC mode

ADD R13, R13,#0X1000 @ Point to top of STACK

CPS #0x12 @ Switch to IRQ mode

LDR R13,=STACK2 @ Point to IRQ stack

CPS #0x13 @ Back to SVC mode

LDR R0,=0x4804C000 @ Base address for GPIO1 registers

ADD R4, R0,#0x190 @ Address of GPIO1\_CLEARDATAOUT register

MOV R7, #0x01E00000 @ Load value to turn off Rotate\_LED on GPIO1\_21,22,23,24

STR R7,[R4] @ Write to GPIO1\_CLEARDATAOUT register

@SET UP FOR GPIO-----------------------------------------------------------------@

ADD R1,R0,#0x0134 @ Get GPIO1\_OE register address

LDR R6,[R1] @ READ current GPIO1 Output Enable register

LDR R7,=0xFE1FFFFF @ Word to enable GPIO1\_21,22,23,24 as output

AND R6,R7,R6 @ Clear bit 12 (MODIFY)

STR R6, [R1] @ WRITE to GPIO1 Output Enable register

@INITIALIZE FALLING EDGE DECTECT-------------------------------------------------@

ADD R1, R0, #0x14C @ Falling edge detect

LDR R2, =0x80000000 @ Detect by GPIO1\_31

LDR R3, [R1] @ Read current value in GPIO1\_FALLING\_EDGE\_DETECT

ORR R3, R3, R2 @ Modify the current value

STR R3, [R1] @ Write modified value back to GPIO1\_FALLINGEDGEDECTECT

ADD R1, R0, #0x34 @ Get address of GPIO1\_IRQSTATUS\_SET0

STR R2, [R1] @ Enable GPIO1\_31 request on GPIO1\_IRQSTATUS\_SET0

@Initialize INTC------------------------------------------------------------------@

@ Reset

LDR R1, =0x48200010 @ INTC\_SYSCONFIG

MOV R2,#2 @ Value for reset

STR R2,[R1] @ Software Reset

@ Enable timer interrupt

LDR R1,=0x482000C8 @ Address of INTC\_MIR\_CLEAR2 register

MOV R2, #0x10 @ Value to unmask INTC INT 68, DMTIMER2, POINTR\_PEND

STR R2, [R1] @ Write to INTC\_MIR\_CLEAR2 register

@ Enable GPIO interrupt

LDR R1, =0x482000E8 @ Address INTC\_MIR\_CLEAR3

MOV R2, #0x00000004 @ Value to unmask INTC INT 98, GPIO1, POINTRPEND1

STR R2, [R1] @ Write to INTC\_MIR\_CLEAR3 register

@SET UP FOR TIMER-----------------------------------------------------------------@

@ Enable Timer 2 Clock

LDR R1, =0x44E00080 @ Address of CM\_PER\_TIMER2\_CLKCTRL

MOV R2, #2 @ Value to enable the timer 2 clock

STR R2, [R1] @ Enable Timer 2 clock

@ Initialize Timer 2

LDR R1, =0x48040000 @ Base address of Timer 2

@ Reset

MOV R2,#1 @ Value for reset

STR R2,[R1,#0x10] @ Store Timer Config ( TIOCP\_CFG)

@ Prime the timer

LDR R2, =0xFFFF8300 @ 1 second count from 0xFFFF8300 to 0xFFFFFFFF

STR R2, [R1,#0x3C] @ Store it into Timer TCRR

LDR R2, =0xFFFF8300 @ 1 second count from 0xFFFF8300 to 0xFFFFFFFF

STR R2, [R1,#0x40] @ Store it into Timer TLDR

@ Start Auto-reload timer

MOV R2,#3 @ Auto-reload, start

STR R2,[R1,#0x38] @ Store it in Time TCLR

@ Initial overflow interrupt

MOV R2,#2 @ Value to enable overflow interrupt

STR R2,[R1,#0x2C] @ Store it Timer Interrupt Enable Set Register

@ Make sure processor IRQ enabRotate\_LED in CPSR

MRS R3, CPSR @ Copy CPSR to R3

BIC R3,#0x80 @ Clear bit 7

MSR CPSR\_c, R3 @ Write back to CPSR

@ Wait for interrupt

MOV R5, #0x0 @ Register hold pointer of status Rotate\_LED

**LOOP:**

NOP

B LOOP

**INT\_DIRECTOR:**

STMFD SP!, {R0-R3, LR} @ Push registers on stack

@ Check if it's GPIO interrupt

LDR R0, =0x482000F8 @ Address of INTC\_PENDING\_IRQ3

LDR R1, [R0] @ Get current value

TST R1, #0x4 @ Test bit 2

BEQ Check\_TIMER\_interrupt @ If interrupt doesn't come from GPIO1

@ Check if it come from GPIO1\_31

LDR R0, =0x4804C02C @ Address of GPIO1\_IRQ\_STATUS

LDR R1, [R0] @ Get current value

TST R1, #0x80000000 @ Test bit 31

BNE BUTTON\_SVC @ If interrupt comes from GPIO1\_31

BEQ PASS\_ON

**Check\_TIMER\_interrupt:**

@ Check that this is the timer interrupt

LDR R0,=0x482000D8 @ Address of INTC-PENDING\_IRQ2 register

LDR R1,[R0] @ Read INTC-PENDING\_IRQ2 register

TST R1,#0x10 @ TEST BIT 4

BEQ PASS\_ON @ Interrupt doesn't come from Timer 2

@ Check that this is the overflow interrupt

LDR R0, =0x48040028 @ Address of Timer 2 IRQ Status

LDR R1,[R0] @ Load value from

CMP R1, #2 @ Check Overflow

BEQ IRQ\_timer @ If overflow -> change state of Rotate\_LED

**PASS\_ON:**

LDMFD SP!, {R0-R3,LR} @ Restore registers

SUBS PC, LR, #4 @ Pass execution on to wait LOOP for now

@---------------------TIMER\_2\_INTERRUPT\_PROCEDURE (OVERFLOW)-------------------@

**IRQ\_timer:**

@ Clear timer interrupts

LDR R1, =0x48040028 @ Timer2\_IRQ Status (IRQSTATUS)

MOV R2, #2 @ Capture, overflow , and match

STR R2, [R1] @ Write 1 to clear interrupt

@ Generate new IRQ generation

LDR R0,=0x48200048 @ Address of INTC\_CONTROL register

MOV R1, #01 @ Value to enable new IRQ generation

STR R1,[R0] @ Write to INTC\_CONTROL register

@ Check value store in FLAG\_button

LDR R0, =FLAG\_button @ Get pointer of FLAG\_button

LDR R2, [R0] @ Get current value in FLAG\_button

CMP R2, #0x0 @ Check current value

@ If flag is 0 --> turn off all Rotate\_LEDs

MOVEQ R7, #0x01E00000 @ Value to turn off all Rotate\_LED

LDREQ R0,=0x4804C190 @ Address of GPIO1\_CLEAR\_DATA\_OUT

STREQ R7,[R0] @ Value to turn off 4 Rotate\_LEDs

MOVEQ R5, #0 @ Reset Pointer register

@ If flag is 1 --> Rotate\_LED rotation

LDRNE R1, =Rotate\_LED @ Get pointer for Rotate\_LED array

LDRNE R3, [R1, R5] @ Get one value in Rotate\_LED array

BNE Rotate\_LED\_ON\_OFF @ Do Rotate\_LED rotation

B DONE

@-----------------PROCEDURE TO TURN ON OR OFF ONE SPECIFIC Rotate\_LED--------------@

**Rotate\_LED\_ON\_OFF:**

LDR R0, =FLAG\_timer @ Get pointer of FLAG\_timer

LDR R2, [R0] @ Get current value

CMP R2, #0x1 @ Check FLAG\_timer

BEQ Timer\_LED\_Off @ if equals 0 --> Timer for LED off

@ ELSE

LDR R0,=0x4804C194 @ Address of GPIO1\_SET\_DATA\_OUT

STR R3,[R0] @ Value to turn on one Rotate\_LED

@ Update FLAG\_Timer for next state will turn off

LDR R0,=FLAG\_timer

MOV R2, #1 @ Assign value 0

STR R2, [R0] @ Store 0 into memory

@ Update pointer for status of Rotate\_LED

CMP R5, #0xC @ Check current value of pointer

MOVEQ R5, #0x0 @ Restore to 0 if the value reach #0xC

ADDNE R5, R5, #4 @ Increament by 4 if the value've not reaches #0xC

B DONE

**Timer\_LED\_Off:**

LDR R0,=0x4804C190 @ Address of GPIO1\_CLEAR\_DATA\_OUT

MOV R3, #0x01E00000

STR R3,[R0] @ Value to turn off one Rotate\_LED

@ Update FLAG\_Timer for next state will turn on

LDR R0,=FLAG\_timer

MOV R2, #0 @ Assign value 1

STR R2, [R0] @ Store 1 into memory

**DONE:**

LDMFD SP!, {R0-R3,LR} @ Restore registers

SUBS PC, LR, #4 @ Pass execution on to wait LOOP for now

@-----------GPIO1 INTERRUPT PROCEDURE (Falling edge detect on GPIO1\_31)-------------@

**BUTTON\_SVC:**

LDR R0,=0x4804C02C @ Load GPIO1\_IRQTATUS register address

LDR R1, =0x80000000 @ Value to turn off GPIO1\_31 interrupt

STR R1, [R0] @ Write to to turn off GPIO1\_31 interrupt

LDR R0,=0x48200048 @ Address of INTC\_CONTROL register

MOV R1,#0x1 @ Value to clear bit 0

STR R1,[R0] @ Write to INTC\_CONTROL register

LDR R0,=0x4804C190

MOV R7, #0x01E00000 @ Load value to turn off Rotate\_LED on GPIO1\_21,22,23,24

STR R7,[R0]

LDR R0,=FLAG\_button @ Load pointer of Flag to check current state

LDR R3,[R0] @ Load value from Flag memory

CMP R3,#0x0 @ Check FLAG\_button

@ Update FLAG\_button

MOVEQ R3,#0x1 @ If current value is 0

STREQ R3,[R0] @ Store 1 to the Check\_Flag for the next check

@ Resotre initialization for pointer and FLAG\_timer

MOVEQ R5, #0 @ Restore pointer Rotate\_LED array

LDREQ R1, =FLAG\_timer @ Get pointer of FLAG\_timer

MOVEQ R2, #0x0 @ Value to restore

STREQ R2, [R1] @ Restore FLAG\_timer

@ ELSE

MOVNE R3,#0x0 @ If current value is 1

STRNE R3,[R0] @ Store 0 to the Check\_Flag for the next check

LDMFD SP!, {R0-R3,LR} @ Restore registers

SUBS PC, LR, #4 @ Pass execution on to wait LOOP for now

**.align** 2

**SYS\_IRQ:** .WORD 0x0 @ Location to store systems IRQ address

**.data**

**FLAG\_button:** **.word** 0x0 @ flag for checking button is press or not

@ FLAG\_button = 1 -> Rotate\_LED turn on

@ FLAG\_button = 0 -> Rotate\_LED turn off

**FLAG\_timer:** **.word** 0x0 @ flag timer for checking timer using LED on or LED off

@ FLAG\_timer = 0 -> Time for LED off

@ FLAG\_timer = 1 -> Time for one LED on

**Rotate\_LED:** **.word** 0x01000000, 0x00800000, 0x00400000, 0x00200000 @ Values for turn in 4 USER Rotate\_LEDS

**STACK1:** .rept 1024

**.word** 0x0000

.endr

**STACK2:** .rept 1024

**.word** 0x0000

.endr

.END