/\*-- constants declaration -------------------------------------------------- \*/

// PAD Multiplexer base address for gpio0, gpio1 and gpio2

#define PADMUX 0x44e10000

// switch & 7-segment gpio offset in pad mux

#define PAD\_OFS\_S1 0x83c

#define PAD\_OFS\_DIG1 0x890

#define PAD\_OFS\_SEGA 0x958

#define PAD\_OFS\_SEGB 0x95c

#define PAD\_OFS\_SEGC 0x980

#define PAD\_OFS\_SEGD 0x820

#define PAD\_OFS\_SEGE 0x824

#define PAD\_OFS\_SEGF 0x828

#define PAD\_OFS\_SEGG 0x82c

// gpio0, gpio1 & gpio2 base address

#define GPIO0 0x44e07000

#define GPIO1 0x4804c000

#define GPIO2 0x481ac000

// gpio register offset

#define OE 0x0134

#define DATAIN 0x0138

#define CLEAR 0x0190

#define SET 0x0194

// press button gpio pin

#define PIN\_S1 15

// 7-segment gpio pin

#define PIN\_DIG1 2

#define PIN\_SEGA 4

#define PIN\_SEGB 5

#define PIN\_SEGC 14

#define PIN\_SEGD 22

#define PIN\_SEGE 23

#define PIN\_SEGF 26

#define PIN\_SEGG 27

// 7-segment bitset

#define DIG1 (1<<PIN\_DIG1)

#define SEGA (1<<PIN\_SEGA)

#define SEGB (1<<PIN\_SEGB)

#define SEGC (1<<PIN\_SEGC)

#define SEGD (1<<PIN\_SEGD)

#define SEGE (1<<PIN\_SEGE)

#define SEGF (1<<PIN\_SEGF)

#define SEGG (1<<PIN\_SEGG)

#define SEG\_ALL (SEGA | SEGB | SEGC | SEGD | SEGE | SEGF | SEGG)

/\*-- implementation of local methods ---------------------------------------- \*/

.text

.align 8

/\*\*

\* method to configure a gpio pin as output

\* @param r0 gpio base address

\* @param r1 pin number

\* @param r2 pad offset

\* @param r3 default outpuz value (0 | 1)

\*/

.global config\_gpio\_out\_pin

config\_gpio\_out\_pin:

nop

// configure default output value

cmp r3, #0

strne r1, [r0, #SET]

streq r1, [r0, #CLEAR]

// configure GPIO pin as out

ldr r3, [r0, #OE]

bic r3, r1

str r3, [r0, #OE]

// configure PAD-Multiplexer as gpio

ldr r3, =PADMUX

mov r1, #0x4f // Value to configurate an output into the PADMUX register

str r1, [r3, r2]

bx lr

/\*\*

\* method to configure a gpio pin as input

\* @param r0 gpio base address

\* @param r1 pin number

\* @param r2 pad offset

\*/

.global config\_gpio\_in\_pin

config\_gpio\_in\_pin:

nop

// configure GPIO pin as input

ldr r3, [r0, #OE]

orr r3, r1

str r3, [r0, #OE]

// configure PAD-Multiplexer as gpio

ldr r3, =PADMUX

mov r1, #0x6f // Value to configurate an intput into the PADMUX register

str r1, [r3, r2]

bx lr

/\*\*

\* method to turn on/off a set of segments

\*/

.global turn\_segments

turn\_segments:

nop

ldr r2, =GPIO0

cmp r1, #1 // if bit segment are activate

ldreq r1, =SET // Set register to activate

ldrne r1, =CLEAR // Clear register to desactivate

str r0, [r2, r1]

bx lr

/\*\*

\* method to display a decimal digit on the 7-segment display

\* r0 digit to display

\*

\*/

.global seg7\_display

seg7\_display:

nop

push {r4, lr}

mov r4, r0 // Move rand value to r4

ldr r0, =SEG\_ALL // Pointer to all segment bit

ldr r1, =0 // 0 for clear

bl turn\_segments // Between all numbers display must desactivate

ldr r0, =table

ldr r0, [r0, r4, lsl #2] // load to r0 which values we want to activate

ldr r1, =1 // 1 for set

bl turn\_segments // Active segmente

pop {r4, pc}

/\*\*

\* method to check if the S1 switch is pressed or not

\* @return r0 if pressed then r0=1 else r0=0

\*/

.global is\_s1\_pressed

is\_s1\_pressed:

nop

ldr r0, =GPIO1 + DATAIN

ldr r0, [r0]

tst r0, #(1<<PIN\_S1) // test if S1 is pressed

movne r0, #0

moveq r0, #1

bx lr

/\*\*

\* method to delay the running of the program

\*/

.global delay

delay:

nop

ldr r0, =50000

1: subs r0, #1

bpl 1b

bx lr

/\*\*

\* ref: https://en.wikipedia.org/wiki/Division\_algorithm

\*

\* --> [IN] = n nu ber

\* --> [IN] = d divisor

\*-->

\*

\* static unsigned modulo (unsigned n, unsigned d)

\* { R0 R0 R1

\* // if (d == 0) return 0;

\* // unsigned q = 0; // quotient

\* unsigned r = 0; // remainder

\*

\* for (int i=31; i>=0; i--) {

\* r <<= 1;

\* if (n & (1<<i)) r |= 1;

\* if (r >= d) {

\* r -= d;

\* // q |= 1<<i;

\* }

\* }

\* return r;

\*

\*/

.global modulo

modulo:

nop

mov r3, #0 // unsigned r = 0 --> remainder

mov r2, #31 // i = 31

mov r12, #1 // used for 1<<i

1: lsl r3, #1 // r <<= 1

tst r0, r12, lsl r2 // r |= n & (1<<i) ? 1 : 0;

orrne r3, #1

cmp r3, r1 // if (r >= d)

subhs r3, r1 // --> r -=d

subs r2, #1 // if i>=0

bge 1b // --> goto 1

mov r0, r3 // return r

bx lr

/\*-- implementation of public methods --------------------------------------- \*/

.global main

main: nop

// print banner...

ldr r0, =banner

bl printf

// initialize the gpio module #0, #1 and #2

mov r0, #0

bl am335x\_gpio\_init

mov r0, #1

bl am335x\_gpio\_init

mov r0, #2

bl am335x\_gpio\_init

// configure gpio pins as output (By default segments are off)

ldr r0, =GPIO2

ldr r1, =DIG1

ldr r2, =PAD\_OFS\_DIG1

ldr r3, =1 // Power on 7-segments

bl config\_gpio\_out\_pin

ldr r0, =GPIO0

ldr r1, =SEGA

ldr r2, =PAD\_OFS\_SEGA

ldr r3, =0

bl config\_gpio\_out\_pin

ldr r0, =GPIO0

ldr r1, =SEGB

ldr r2, =PAD\_OFS\_SEGB

ldr r3, =0

bl config\_gpio\_out\_pin

ldr r0, =GPIO0

ldr r1, =SEGC

ldr r2, =PAD\_OFS\_SEGC

ldr r3, =0

bl config\_gpio\_out\_pin

ldr r0, =GPIO0

ldr r1, =SEGD

ldr r2, =PAD\_OFS\_SEGD

ldr r3, =0

bl config\_gpio\_out\_pin

ldr r0, =GPIO0

ldr r1, =SEGE

ldr r2, =PAD\_OFS\_SEGE

ldr r3, =0

bl config\_gpio\_out\_pin

ldr r0, =GPIO0

ldr r1, =SEGF

ldr r2, =PAD\_OFS\_SEGF

ldr r3, =0

bl config\_gpio\_out\_pin

ldr r0, =GPIO0

ldr r1, =SEGG

ldr r2, =PAD\_OFS\_SEGG

ldr r3, =0

bl config\_gpio\_out\_pin

// configure gpio pins as input

// dice playing main loop

/\*Next exemple in C

seg7\_dsiplay(0);

while(true) {

if (!is\_s2\_pressed()) continue;

int digit = (rand();

digit = modulo(digit, 6) + 1;

seg7\_dispplay(digit);

delay();

}

\*/

ldr r0, =GPIO1

ldr r1, =PIN\_S1

ldr r2, =PAD\_OFS\_S1

bl config\_gpio\_in\_pin

mov r0, #0

bl seg7\_display

1: nop

bl is\_s1\_pressed

cmp r0, #0

beq 1b

bl rand // Intern random function

mov r1, #6

bl modulo

add r0, #1

bl seg7\_display

bl delay

b 1b

// String definition used for message outputs (printf)

.section .rodata

banner: .ascii "\n"

.ascii "HEIA-FR - Embedded Systems 1 Laboratory\n"

.ascii "An introduction the ARM's assembler language\n"

.ascii "--> a dice roller on Beaglebone Black HEIA-FR extension board\n"

.asciz "\n"

/\* 7-segment: segment definition

+-- SEGA --+

| |

SEGF SEGB

| |

+-- SEGG --+

| |

SEGE SEGC

| |

+-- SEGD --+

\*/

// Table of segment to make numbers

.align 2

table:

.long (SEGA+SEGB+SEGC+SEGD+SEGE+SEGF) //0

.long (SEGB + SEGC) //1

.long (SEGA+SEGB+SEGG+SEGE+SEGD) //2

.long (SEGA + SEGB + SEGG + SEGC + SEGD) //3

.long (SEGF + SEGG + SEGB + SEGC) //4

.long (SEGA + SEGF + SEGG + SEGC + SEGD) //5

.long (SEGA + SEGF + SEGG + SEGE + SEGD + SEGC) //6