link.ld

```
 /*Mit ENTRY wird der Einstiegspunkt an _start definiert
SECTIONS legt die Bereiche der Segmente im Speicher fest*/

 OUTPUT_ARCH( "riscv" )
 ENTRY( _start )
 SECTIONS
 {   /* text: test code section */
    . = 0x00000000
    .text : { *(.text) }
    /* data: Initialized data segment */
    .data : { *(.data) }
    /* End of uninitalized data segment */
    _end = .;
 }

```

crt0.s

```
#Setzt Stackpointer und ruft die Funktion main auf
.section .text
.global _start
_start:
# reset vector at 0x0
.=0x0
j _init

# interrupt handler
.=0x10
_interrupt:
# for now just do an endless loop
j _interrupt
_init:
# set up stack pointer
li sp,4096
# call main function
jal ra,main
# back to start
j _start

```

Makefile

```
#Sourcefile: $<; Generated File: $@
#Objektdatei .o erzeugen
%.o: %.c
    riscv64-unknown-elf-gcc  -march=rv32i -mabi=ilp32 -static -nostdlib
    -fno-builtin-printf -Os -fPIC -c $<

#elf-Datei erzeugen und crt0.s und Linkerskript link.ld einbinden
%.elf: %.o
    riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -static -nostdlib
    -fno-builtin-printf -Os -fPIC -fdata-sections -ffunction-sections -o $@
    crt0.s $< -Tlink.ld -Xlinker --gc-sections

#Umwandeln in bin-Datei
%.bin: %.elf
    riscv64-unknown-elf-objcopy -O binary $< $@

#Umwandeln in dat-Datei
%.dat: %.bin
    hexdump -v -e '1/1 "%02x" "\n"' $< > $@

```

led.c

```c

#include <stdint.h>
#include <stdbool.h>
#define reg_leds (*(volatile uint8_t*)0xFFFFFFE2)

void main() {

    uint32_t led = 0;

    while (1) {
    //32 Bit werden hochgezählt, Bit 17-21 werden an den Leds angezeigt
        reg_leds = (led >> 16) & 0xFF;
        led = led  + 1;
    }
}

```

blink-test.s

```
1  #Register t0 wird auf 0 gesetzt.
2  #In einer Schleife wird dem Wert 1 dazu addiert und
3  # um 16 Stellen nach rechts geshiftet in Register t1 geschrieben.
4  #Danach wird der Wert an der Speicheradresse der Leds gespeichert.
5
6  #include "devices.h"
7
8  .section .text
9
10 .global _start
11
12 main:
13      li t0,0
14
15 loop:
16      addi t0,t0,1
17      srli t1,t0,16
18      sb t1,DEV_LED(zero)
19
20 j loop
21
22 .size   main, .-main
23
24
```

button.c

```c
/*
Von 8 Bit Daten sind 5 Bit für die 5 Buttons bzw. 5 Leds
Wird ein Button gedrückt, leuchtet die jeweilige Led
*/

#define reg_leds (*(volatile uint8_t*)0xFFFFFFE2)  //Adresse Leds
#define reg_button (*(volatile uint8_t*)0xFFFFFFE0) //Adresse Buttons

void main() {
    uint8_t btn0 = reg_button;
    while (1) {
         btn0 = reg_button; //Einlesen der Buttons Bit0=Buttton1, Bit1=Button2
         , etc.
         reg_leds = btn0; //Ausgabe an Leds Bit0=Led1, Bit1=Led2, etc.
    }
}

```

gpio.c

```c
/*
Testprogramm für GPIO:
Es werden gpio0 und gpio1 als Output konfiguriert. In der Schleife werden die Buttons eingelesen und an die Leds und Gpio0 ausgegeben. Gpio1 bekommt das Eingelesene invertiert.
*/
#include <stdint.h>
#include <stdbool.h>

#define reg_leds (*(volatile uint8_t*)0xFFFFFFE2)
#define reg_button (*(volatile uint8_t*)0xFFFFFFE0)
#define reg_gpio0_dir (*(volatile uint8_t*)0xFFFFFE01)
#define reg_gpio0_value (*(volatile uint8_t*)0xFFFFFE00)
#define reg_gpio1_dir (*(volatile uint8_t*)0xFFFFFE03)
#define reg_gpio1_value (*(volatile uint8_t*)0xFFFFFE02)


void main() {

    uint8_t btn0 = reg_button;
    reg_gpio0_dir = (uint8_t)0b11111111;
    reg_gpio1_dir = (uint8_t)0b11111111;

    while (1) {
        btn0 = reg_button;
        reg_leds = btn0;
        reg_gpio0_value = btn0;
        reg_gpio1_value = ~btn0;
    }
}

```

switch.c

```c
/*
Testprogramm für Switches:
Die beiden Switches werden in einer Schleife eingelesen.
Wenn Switch1/2 geschlossen ist, gehen die ersten/letzten 4 Leds an.
Wenn beide Switches geschlossen sind, gehen alle 8 Leds an.
*/

#include <stdint.h>
#include <stdbool.h>

#define reg_leds (*(volatile uint8_t*)0xFFFFFFE2)
#define reg_switches (*(volatile uint8_t*)0xFFFFFFE1)


void main() {

    //1.Bit von reg_switches = Switch1
    //2.Bit von reg_switches = Switch2
    uint8_t sw1 = reg_switches & 0x1;
    uint8_t sw2 = reg_switches>>1 & 0x1;

    while (1) {

    sw1 = reg_switches & 0x1;
        sw2 = reg_switches>>1 & 0x1;

    if(sw1 && sw2==0){
        reg_leds = 0x0F;

    }else if(sw2 && sw1==0){
        reg_leds = 0xF0;

    }else if(sw1 && sw2){
        reg_leds = 0xFF;
    }
    else {
        reg_leds = 0;
    }

    }
}
```