Homework1

1. **What is the range of voltages that represent logic low?**
   - 1 - 1.3V
2. **What is the range of voltages that represent logic high?**
   - 2 - 5V
3. **What is the difference between positive and negative logic?**
   - 正逻辑：高电平表示逻辑1，低电平表示逻辑0。
   - 负逻辑：高电平表示逻辑0，低电平表示逻辑1。
   - 正逻辑和负逻辑表示逻辑0和1的方式是相反的。
   - 具体实现：在正逻辑中，外部的电阻是一个下拉电阻，如果开关闭合，就会给输入端口一个高电平，如果开关打开，就会给输入端口一个低电平（接地）。 在负逻辑中，外部的电阻是一个上拉电阻，如果开关闭合，就会给输入端口一个低电平（接地），如果开关打开，就会给输入端口一个高电平。
4. **What is the difference between volatile and nonvolatile memory?**
   - volatile memory：断电后Volatile memory中的数据就会丢失
   - nonvolatile memory：断电后nonvolatile memory仍然保留不会丢失
5. **What is flash?**
   - Flash是一种非易失性内存，在没有电流供应的条件下也能够长久地保持数据，其存储特性相当于硬盘，这项特性正是闪存得以成为各类便携型数字设备的存储介质的基础。
   - Flash属于广义的EEPROM（电可擦可编程只读存储器，一种掉电后数据不丢失的存储芯片），因为它也是电擦除的ROM。但是为了区别于一般的按字节为单位的擦写的EEPROM，其称其为Flash。
6. **What is a pin? What is a port?**
   - Pin指的是引脚，是从芯片内部电路引出的与外围电路的接线。
   - Port指的是端口，可以认为是设备与外界通讯交流的出口，端口是按共同功能分组的引脚集合，也就是一个Port中可能会有多个Pin。
7. **What does real-time mean?**
   - 实时意味着系统对事件的响应时间总是小于一个界限。就是系统能够及时响应外部事件的请求，在规定时间内完成对该事件的处理。要求系统在处理任务时，不仅要满足逻辑的正确性，还要满足时间约束条件。
8. **How much RAM and ROM does our microcontroller have?**

   TM4C123

   - RAM：32kb
   - ROM：256 kb
9. **How do we change the following Program to run using Port A(set PA7 out)?**

   程序改为

```
void PortA_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000001;     // 1) activate clock for Port A
  delay = SYSCTL_RCGC2_R;           // allow time for clock to start
  GPIO_PORTA_LOCK_R = 0x4C4F434B;   // 2) unlock GPIO Port A
  GPIO_PORTA_CR_R |= 0x80;          // allow changes to PA7
  GPIO_PORTA_AMSEL_R &= ~0x80;      // 3) disable analog on PA7
  GPIO_PORTA_PCTL_R &= ~0xF0000000; // 4) PCTL GPIO on PA7
  GPIO_PORTA_DIR_R |= 0x80;         // 5) PA7 out
  GPIO_PORTA_AFSEL_R = 0x00;        // 6) disable alt funct on PA7-0
  GPIO_PORTA_PUR_R = 0x00;          // enable pull-up on none
  GPIO_PORTA_DEN_R |= 0x80;         // 7) enable digital I/O on PA7
}
```

10. **The base address for Port A is 0x4000.4000. If we want to read and write all 8 bits of this port, the constants will add up to 0x03FC. In other words, read and write operations to GPIO_PORTA_DATA_R will access all 8 bits of Port A. If we are interested in just bit 5 of Port A, address for Port PA5, how we define this in C and in assembly? If we define PA5 as bit address, does the following code modify the other 7 bits of Port A?**

    For example: PA5 = 0x20; // make PA5 high

    - C

    ```
    #define PA5   (*((volatile unsigned long *)0x40004080))
    ```

    - assembly

    ```
    PA5 EQU 0x40004080
    ```

    上述例子不修改端口A的其他7位

11. **Assume an LED is attached to Port F bit 2. Write toggle the LED operations in C and in assembly will create a bit-specific address constant to access just PF2.**

    - 代码确保能在单片机上运行，完整代码见附件
    - C

    ```
    #define PF2 (*((volatile uint32_t *)0x40025010));
    void Toggle(void){
      PF2 ^= 0x04;  // toggle LED
    }
    ```

    - assembly

```
PF2 EQU 0x40025010
Toggle
    LDR R1, = PF2       ; pointer to PF2 (or use LDR R1, = 0x40025010)
    LDR R0, [R1]        ; read PF2
    EOR R0, R0, #0x04   ; R0 = R0 ^ 0x04
    STR R0, [R1]        ; write to PF2
    BX  LR
```

附件

```
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#define GPIO_LOCK_KEY 0x4C4F434B    // Unlocks the GPIO_CR register
#define PF2 (*((volatile uint32_t *)0x40025010))
#define SYSCTL_RCGC2_GPIOF 0x00000020  // port F Clock Gating Control
void PortF_Init(void){ volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;     // 1) activate clock for Port F
  delay = SYSCTL_RCGC2_R;           // allow time for clock to start
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // 2) unlock GPIO Port F
  GPIO_PORTF_CR_R = 0x04;           // allow changes to PF2
  // only PF0 needs to be unlocked, other bits can't be locked
  GPIO_PORTF_AMSEL_R = 0x00;        // 3) disable analog on PF
  GPIO_PORTF_PCTL_R = 0x00000000;   // 4) PCTL GPIO on PF2
  GPIO_PORTF_DIR_R = 0x04;          // 5) PF2 out
  GPIO_PORTF_AFSEL_R = 0x00;        // 6) disable alt funct on PF7-0
  GPIO_PORTF_PUR_R = 0x00;          // enable pull-up on none
  GPIO_PORTF_DEN_R = 0x04;          // 7) enable digital I/O on PF2
}
void Toggle(void){
  PF2 ^= 0x04;  // toggle LED
}
int main(void) {
  PortF_Init();
  while (1) {
    Toggle();
    uint32_t i;                 //delay
    for(i=0;i<500000;i++);
  }
}
```

```
GPIO_PORTF_DATA_R  EQU 0x400253FC
GPIO_PORTF_DIR_R   EQU 0x40025400
GPIO_PORTF_AFSEL_R EQU 0x40025420
GPIO_PORTF_PUR_R   EQU 0x40025510
GPIO_PORTF_DEN_R   EQU 0x4002551C
GPIO_PORTF_LOCK_R  EQU 0x40025520
GPIO_PORTF_CR_R    EQU 0x40025524
GPIO_PORTF_AMSEL_R EQU 0x40025528
```

```
GPIO_PORTF_PCTL_R  EQU 0x4002552C
GPIO_LOCK_KEY      EQU 0x4C4F434B  ; Unlocks the GPIO_CR register
SYSCTL_RCGCGPIO_R  EQU   0x400FE608

        AREA    |.text|, CODE, READONLY, ALIGN=2
        THUMB
        EXPORT  Start

Start
    BL  PortF_Init                  ; initialize input and output pins of Port F
loop
    LDR R0, = FIFTHSEC              ; R0 = FIFTHSEC (delay 0.2 second)
    BL  delay                       ; delay at least (3*R0) cycles
    BL  Toggle                      ; turn all of the LEDs on
    B   loop

;------------delay------------
; Delay function for testing, which delays about 3*count cycles.
; Input: R0  count
; Output: none
ONESEC             EQU 5333333      ; approximately 1s delay at ~16 MHz clock
QUARTERSEC         EQU 1333333      ; approximately 0.25s delay at ~16 MHz clock
FIFTHSEC           EQU 1066666      ; approximately 0.2s delay at ~16 MHz clock
delay
    SUBS R0, R0, #1                 ; R0 = R0 - 1 (count = count - 1)
    BNE delay                       ; if count (R0) != 0, skip to 'delay'
    BX  LR                          ; return

;------------PortF_Init------------
; Initialize GPIO Port F for negative logic switches on PF0 and
; PF4 as the Launchpad is wired.  Weak internal pull-up
; resistors are enabled, and the NMI functionality on PF0 is
; disabled.  Make the RGB LED's pins outputs.
; Input: none
; Output: none
; Modifies: R0, R1, R2
PortF_Init
    LDR R1, =SYSCTL_RCGCGPIO_R      ; 1) activate clock for Port F
    LDR R0, [R1]
    ORR R0, R0, #0x20               ; set bit 5 to turn on clock
    STR R0, [R1]
    NOP
    NOP                             ; allow time for clock to finish
    LDR R1, =GPIO_PORTF_LOCK_R      ; 2) unlock the lock register
    LDR R0, =0x4C4F434B             ; unlock GPIO Port F Commit Register
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_CR_R        ; enable commit for Port F
    MOV R0, #0x04                   ; 1 means allow access
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_AMSEL_R     ; 3) disable analog functionality
    MOV R0, #0                      ; 0 means analog is off
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_PCTL_R      ; 4) configure as GPIO
    MOV R0, #0x00000000             ; 0 means configure Port F as GPIO
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_DIR_R       ; 5) set direction register
    MOV R0, #0x04                   ;  PF2 output
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_AFSEL_R     ; 6) regular port function
    MOV R0, #0                      ; 0 means disable alternate function
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_PUR_R       ; pull-up resistors for PF4,PF0
    MOV R0, #0x00                   ; enable weak pull-up on none
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_DEN_R       ; 7) enable Port F digital port
    MOV R0, #0x04                   ; 1 means enable digital I/O PF2
    STR R0, [R1]
    BX  LR

;------------Toggle------------
; Set the output state of PF2.
; Input: R0  new state of PF
; Output: none
; Modifies: R1
Toggle
    LDR R1, = 0x40025010            ; pointer to PF2
    LDR R0, [R1]                    ; read PF2
    EOR R0, R0, #0x04               ; R0 = R0 ^ 0x04
    STR R0, [R1]                    ; write to PF2
    BX  LR

    ALIGN                           ; make sure the end of this section is aligned
    END
```