ZHEJIANG UNIVERSITY　COLLEGE OF INFORMATION SCIENCE AND ELECTRONICS ENGINEERING

HOMEWORK #2

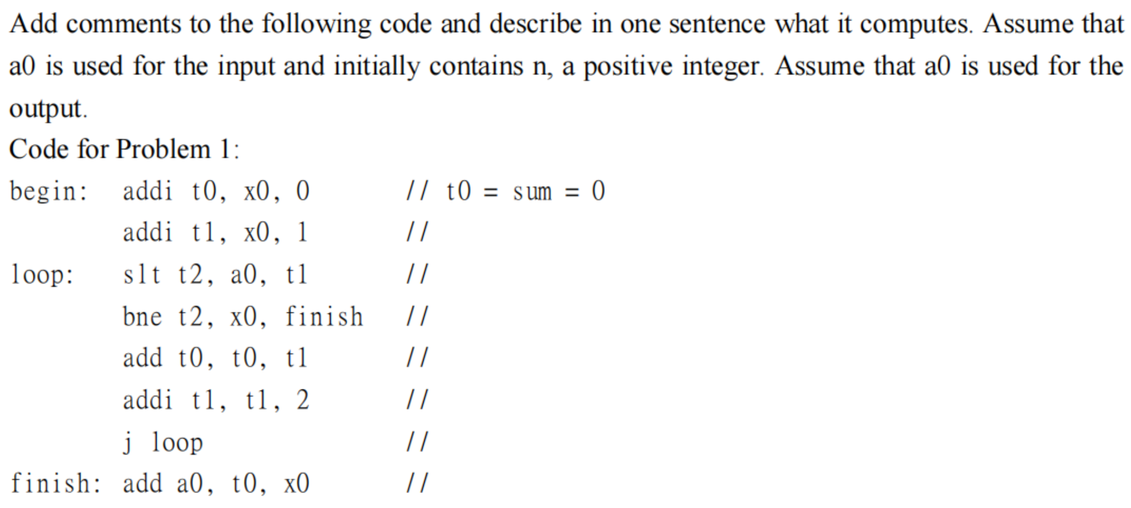
Computer Organization and Design

Name:　　　　　　Student ID:

Major: Electronic Science and Technology

Date: 2024 年 11 月 17 日

Problem 1.



Add comments to the following code and describe in one sentence what it computes. Assume that a0 is used for the input and initially contains n, a positive integer. Assume that a0 is used for the output.

Code for Problem 1:

```
begin:  addi t0, x0, 0       // t0 = sum = 0
        addi t1, x0, 1       //
loop:   slt t2, a0, t1       //
        bne t2, x0, finish   //
        add t0, t0, t1       //
        addi t1, t1, 2       //
        j loop               //
finish: add a0, t0, x0       //
```

图 1　pro1

Answer:　The code calculates the sum of all odd numbers from 1 to n.

Listing 1　Code for Problem 1

```
begin: addi t0, x0, 0     // t0 = sum = 0
addi t1, x0, 1            // t1 = i = 1
loop: slt t2, a0, t1      // if i < n
bne t2, x0, finish        // goto finish
add t0, t0, t1            // sum += i
addi t1, t1, 2            // i += 2
j loop                    // goto loop for next iteration
finish: add a0, t0, x0    // a0 = sum
```

Problem 2. Write a loop that reverses order of the bits of an 8-bit number in s0 and stores the result in s1. RISC-V registers have 4 bytes, but in this problem, the higher 24 bits of s0 are always 0. For example, if the lower 8 bits of s0 are 00001101, the lower 8 bites s1 should be 10110000.

Answer:

Listing 2 Code for Problem 2

```
_start:
    li t0, 8            # 设置循环计数器 t0 为 8
    li s1, 0            # 将 s1 初始化为 0
    li t1, 1            # t1 用于提取 s0 中的每一位
    li t2, 128          # t2 用于设置 s1 中的每一位

reverse_loop:
    beqz t0, end_loop # 如果 t0 为 0，跳转到 end_loop
    and t3, s0, t1      # 提取 s0 中的当前位
    beqz t3, skip_set # 如果当前位为 0，跳过设置
    or s1, s1, t2       # 设置 s1 中的当前位

skip_set:
    srli s0, s0, 1      # 右移 s0
    srli t2, t2, 1      # 右移 t2
    addi t0, t0, -1     # 递减 t0
    j reverse_loop      # 跳转到 reverse_loop

end_loop:
    # 结束程序
    li a7, 10           # ecall 10 系统调用结束程序
    ecall
```

Problem 3. Assume we have an array in memory that contains int* arr = 1,2,3,4,5,6,0. Let the values of arr be a multiple of 4 and stored in register s0. What do the snippets of RISC-V code do? Assume that all the instructions are run one after the other in the same context.

Listing 3 Code for Problem 2

```
a)  lw t0, 8(s0)
b)  slli t1, t0, 2
    add t2, s0, t1
    lw t3, 0(t2)
    addi t3, t3, 1
    sw t3, 0(t2)
c)  lw t0, 16(s0)
    xori t0, t0, 0xFFF
    addi t0, t0, 1
```

Answer :

- Snippet a: t0=arr[2]=3;
  - `lw t0, 8(s0)`
    t0=arr[2]=3
- Snippet b: arr[3]++;
  - `slli t1, t0, 2`
    t1=t0«2, so `t1` will be set to `3 << 2 = 12`.
  - `add t2, s0, t1`
    t2=s0+t1=s0+12
  - `lw t3, 0(t2)`
    equivalent to `lw t3, 12(s0)`, t3=arr[3]=4
  - `addi t3, t3, 1`
    t3=t3+1=5
  - `sw t3, 0(t2)`
    equivalent to `sw t3, 12(s0)`, arr[3]=5
- Snippet c:t0=-arr[4];
  - `lw t0, 16(s0)`
    t0=arr[4]=5
  - `xori t0, t0, 0xFFF`
    t0=t0$\oplus$0xFFF=5$\oplus$0xFFF=0xFFA

- `addi t0, t0, 1`
  t0=t0+1=0xFFB

Problem 4. Write a function sumSquare in RISC-V that, when given an integer n, returns the summation below. If n is not positive, then the function returns 0.

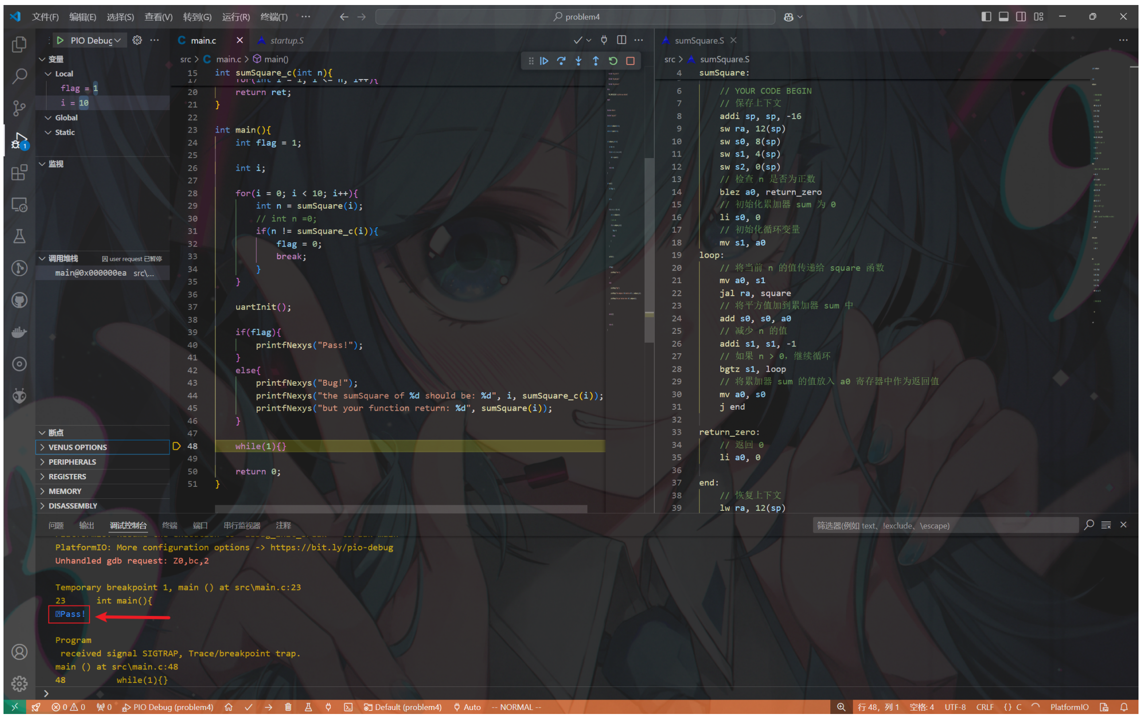
$$\Sigma_{i=1}^{n} i^2 \tag{1}$$

For this problem, you are given a RISC-V function called square that takes in an integer and returns its square. Implement sumSquare using square as a subroutine

Answer:

Listing 4 Code for Problem 2

```
.globl sumSquare
.text
sumSquare:
    // YOUR CODE BEGIN
    // 保存上下文
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    // 检查 n 是否为正数
    blez a0, return_zero
    // 初始化累加器 sum 为 0
    li s0, 0
    // 初始化循环变量
    mv s1, a0
loop:
    // 将当前 n 的值传递给 square 函数
    mv a0, s1
    jal ra, square
    // 将平方值加到累加器 sum 中
    add s0, s0, a0
    // 减少 n 的值
    addi s1, s1, -1
    // 如果 n > 0, 继续循环
    bgtz s1, loop
    // 将累加器 sum 的值放入 a0 寄存器中作为返回值
    mv a0, s0
    j end
return_zero:
    // 返回 0
    li a0, 0
end:
    // 恢复上下文
    lw ra, 12(sp)
    lw s0, 8(sp)
    lw s1, 4(sp)
    lw s2, 0(sp)
    addi sp, sp, 16
    // YOUR CODE END
    ret
.end
```



```
int sumSquare_c(int n){
    return ret;
}

int main(){
    int flag = 1;

    int i;

    for(i = 0; i < 10; i++){
        int n = sumSquare(i);
        // int n =0;
        if(n != sumSquare_c(i)){
            flag = 0;
            break;
        }
    }

    uartInit();

    if(flag){
        printfNexys("Pass!");
    }
    else{
        printfNexys("Bug!");
        printfNexys("the sumSquare of %d should be: %d", i, sumSquare_c(i));
        printfNexys("but your function return: %d", sumSquare(i));
    }

    while(1){}

    return 0;
}
```

```
sumSquare:
    // YOUR CODE BEGIN
    // 保存上下文
    addi sp, sp, -16
    sw ra, 12(sp)
    sw s0, 8(sp)
    sw s1, 4(sp)
    sw s2, 0(sp)
    // 检查 n 是否为正数
    blez a0, return_zero
    // 初始化累加器 sum 为 0
    li s0, 0
    // 初始化循环变量
    mv s1, a0
loop:
    // 将当前 n 的值传递给 square 函数
    mv a0, s1
    jal ra, square
    // 将平方值加到累加器 sum 中
    add s0, s0, a0
    // 减少 n 的值
    addi s1, s1, -1
    // 如果 n > 0, 继续循环
    bgtz s1, loop
    // 将累加器 sum 的值放入 a0 寄存器中作为返回值
    mv a0, s0
    j end

return_zero:
    // 返回 0
    li a0, 0

end:
    // 恢复上下文
    lw ra, 12(sp)
```

```
PlatformIO: More configuration options -> https://bit.ly/pio-debug
Unhandled gdb request: Z0,bc,2

Temporary breakpoint 1, main () at src\main.c:23
23      int main(){
Pass!

Program
 received signal SIGTRAP, Trace/breakpoint trap.
main () at src\main.c:48
48          while(1){}
```

图 2 result