計算機架構\_CH2\_HW2

**資工三 B0329044 蔡宛芝**

2.22

For the following C statement, write a minimal sequence of MIPS assembly instructions that does the identical operation. Assume ＄t1 = A, ＄t2 = B, and ＄s1 is the base address of C.

A = C[0] << 4;

**Ans：**

**lw $t1 , 0 ($s1)**

**sll $t1 , $t1 , 4**

2.23

Assume ＄t0 holds the value 0x00101000. What is the value of ＄t2 after the following instructions?

slt ＄t2, ＄0, ＄t0

bne ＄t2, ＄0, ELSE

j DONE

ELSE: addi ＄t2, ＄t2, 2

DONE:

**Ans：**

**$t2=3**

2.27

Translate the following C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers ＄s0, ＄s1, ＄t0, and ＄t1, respectively. Also, assume that register ＄s2 holds the base address of the array D.

for(i=0; i<a; i++)

for(j=0; j<b; j++)

D[4\*j] = i + j;

**Ans：**

**add $t0, $0, $0 # i = 0**

**L1: slt $t2, $t0, $s0 # i < a**

**beq $t2, $0, Exit # $t2 == 0, go to Exit**

**add $t1, $0, $0 # j = 0**

**L2: slt $t2, $t1, $s1 # j < b**

**beq $t2, $0, L3 # if $t2 == 0, go to L3**

**add $t2, $t0, $t1 # i+j**

**sll $t4, $t1, 2 # $t4 = 4\*j**

**add $t3, $t4, $s2 # $t3 = &D[4\*j]**

**sw $t2, 0($t3) # D[4\*j] = i+j**

**addi $t1, $t1, 1 # j = j+1**

**j L2**

**L3: addi $t0, $t0, 1 # i = i+1**

**j L1**

**Exit:**

2.31

Implement the following C code in MIPS assembly. What is the total number of MIPS instructions needed to execute the function?

int fib(int n) {

if (n==0)

return 0;

else if (n==1)

return 1;

else

return fib(n-1) + fib(n-2);

}

**Ans：**

**fib:**

**addi $sp, $sp, -12 # allocate stack frame of 12 bytes**

**sw $a0, 8($sp) # save n**

**sw $ra, 4($sp) # save return address**

**sw $s0, 0($sp) # save $s0**

**slti $t0, $a0, 2 # fib(i) = i for i = 0, 1**

**beq $t0, $0, else**

**add $v0, $a0, $0 # $v0 = 0 or 1**

**j exit # go to exit**

**else:**

**addi $a0, $a0, -1 # fib(n-1)**

**jal fib # recursive call**

**add $s0, $v0, $0**

**addi $a0, $a0, -1 # fib(n-2)**

**jal fib # recursive call**

**add $v0, $v0, $s0**

**exit:**

**lw $a0, 8($sp) # restore $a0**

**lw $ra, 4($sp) # restore return address**

**lw $s0, 0($sp) # restore $s0**

**addi $sp, $sp, 12 # free stack frame**

**jr $ra # return to caller**