# Chapter 2

1. **2.10 Assume that registers x5 and x6 hold the values 0x8000000000000000 and 0xD000000000000000, respectively.**
2. The value of x30 for the following assembly code is 0x\_5000000000000000\_\_\_\_\_\_\_\_\_\_\_

add x30, x5, x6

1. Is the result in x30 the desired result, or has there been overflow? ( A )

A. overflow B. no overflow

1. For the contents of registers x5 and x6 as specified above, The value of x30 for the following assembly code is 0x B000000000000000\_\_\_\_\_\_\_\_\_\_\_\_

sub x30, x5, x6

1. Is the result in x30 the desired result, or has there been overflow? ( B )

A. overflow B. no overflow

1. For the contents of registers x5 and x6 as specified above, The value of x30 for the following assembly code is 0x\_\_D000000000000000\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

add x30, x5, x6

add x30, x30, x5

1. Is the result in x30 the desired result, or has there been overflow?( A )

A. overflow B. no overflow

1. **2.22 Suppose the program counter (PC) is set to 0x20000000.**
2. What range of addresses can be reached using the RISC-V jump-and-link (jal) instruction? (In other words, what is the set of possible values for the PC after the jump instruction executes?)

The range is [0x\_ 1FF00000\_\_\_\_\_\_, 0x\_ 200FFFFE\_\_\_\_\_](low to high)

1. What range of addresses can be reached using the RISC-V branch if equal (beq) instruction? (In other words, what is the set of possible values for the PC after the branch instruction executes?)

The range is [0x\_\_ 1FFFF000\_\_\_\_\_\_\_, 0x\_\_20000FFE\_\_\_\_]( from low to high)

1. **2.29 Implement the following C code in RISC-V assembly. Hint: Remember that the stack pointer must remain aligned on a multiple of 16.**

int fib(int n){

if (n==0)

return 0;

else if (n == 1)

return 1;

else

return fib(n−1) + fib(n−2);

}

fib:

beq x10, x0, finish //if n==0 return 0

addi x5, x0, 1

beq x10, x5, finish //if n==1 return 1

addi x2, x2, -16

sd x1 ,0(x2) //save x1 on stack

sd x10, 8(x2) //save x10 on stack

addi x10, x10, -1 //n-1

jal x1, fib //fib(n-1)

ld x5, 8(x2) //x5 get n

sd x10, 8(x2) //push fib(n-1) onto the stack

addi x10, x5, -2 //n-2

jal x1,fib //fib(n-2)

ld x5,8(x2) //x5 get fib(n-1)

add x10, x10, x5 //fib(n) = fib(n-1) + fib(n-2)

ld x1, 0(x2) //return saved rd

addi x2, x2, 16 //pop back

finish:

jalr x0,0(x1) //Return to caller

1. **2.5 Show how the value 0xabcdef12 would be arranged in memory of a little-endian and a big-endian machine. Assume the data are stored starting at address 0 and that the word size is 4 bytes.**

**a big-endian machine**

address data

3 12

2 ef

1 cd

0 ab

**a little-endian machine**

address data

3 ab

2 cd

1 ef

0 12

1. **2.12 Provide the instruction type and assembly language instruction for the following binary value: 0000 0000 0001 0000 1000 0000 1011 0011two**

1) the type of the instruction is( A )

A. R B. I C. B D. J

2) the instruction is (A )

A. add x1,x1,x1 B. addi x1,x1,x1 C. beq x1,x2,4 D.jalr x0,0(x2)

# chapter3

1. **Assume decimal integers 185 and 122 are unsigned 8-bit integers, their bit patterns are A and B, now A and B represent signed 8-bit decimal integers stored in sign-magnitude format, Calculate A + B.**

1) The result is \_65\_(in decimal)

2) ( C ) is in this calculation.

A. overflow B. underflow C. neither

1. **3.20 Given the bit pattern 0x0C000000, if it is a two’s complement integer ,its decimal value is \_201326592\_\_\_\_\_\_\_\_\_, and an unsigned integer is \_201326592 \_\_\_\_\_\_\_\_**
2. **3.22 What decimal number does the bit pattern 0x0C000000 represent if it is a floating point number? Use the IEEE 754 standard. B**

**A. 1.0\*2-101 B. 1.0\*2-103 C. 1.05\*2-101 D. 1.05\*2-103**

1. **3.23 The binary representation of the decimal number 63.25 assuming the IEEE 754 single precision format is 0x\_427D0000\_\_\_\_\_\_in hex.**