**Problem 1:**

Assume address in memory of ‘A[0]’, ‘B[0]’ and ‘C[0]’) are stored in Registers x27, x30, x31. Assume values of variables f, g, h, i, and j are assigned to registers x5, x6, x7, x28, x29 respectively

*Write down RISC V Instruction(s) to*

(a) Load Register x5 *with content*of A[10]

lw x5, 40(x27)

(b) Store contents of Register x5 into A[17]

sw x5, 68(x27)

(c) add 2 operands: one in x5 - a register, the other in in Register x6. Assume result of operation to be stored in register x7

add x7, x5, x6

(d) copy contents at one memory location to another: C[g] = A[i+j+31]

add x28, x28, x29

addi x28, x28, 31

slli x28, x28, 2

add x28, x28, x27

slli x6, x6, 2

add x6, x6, x31

lw x28, 0(x28)

sw x28, 0(x6)

(e) implement in RISC V these line of code in C:

(i) f = g - A[B[9]]

lw x8, 36(x30)

slli x8, x8, 2

add x8, x8, x27

lw x8, 0(x8)

sub x5, x6, x8

(ii) f = g - A[C[8] + B[4]]

lw x30, 16(x30)

lw x31, 32(x31)

add x8, x30, x31

slli x8, x8, 2

add x8, x8, x27

lw x8, 0(x8)

sub x5, x6, x8

(iii) A[i] = B[2i+1], C[i] = B[2i]

add x26, x28, x28

addi x26, x26, 1

slli x26, x26,2

add x26, x26, x30

slli x25, x28, 2

add x25, x25, x27

lw x24, 0(x26)

sw x24, 0(x25)

---------------------------------------------------

addi x26, x26, -4

slli x23, x28, 2

add x23, x23, x31

lw x24, 0(x26)

sw x24, 0(x23)

(iv) A[i] = 4B[i-1] + 4C[i+1]

addi x26, x28, -1

slli x26, x26, 2

add x26, x26, x30

addi x25, x28, 1

slli x25, x25, 2

add x25, x25, x31

slli x24, x28, 2

add x24, x24, x27

lw x23, 0(x26)

slli x23, x23, 2

lw x22, 0(x25)

slli x22, x22, 2

add x23, x23, x22

sw x23, 0(x24)

(v) f = g - A[C[4] + B[12]]

lw x30, 48(x30)

lw x31, 16(x31)

add x24, x30, x31

slli x24, x24, 2

add x4, x27, x24

lw x4, 0(x4)

sub x5, x6, x4

**Problem 2:**

Assume the following register contents:

x5 = 0x00000000AAAAAAAA, x6 = 0x1234567812345678

**a.** For the register values shown above, what is the value of x7 for the following sequence of instructions?

**srli** x7, x5, 16

x7 = 0x000000000000AAAA

addi x7, x7, -128

X7 = 00001010101000101010

srai x7, x7, 2

x7 = 00000010101010001010

and x7, x7, x6

x7 = 10 0000 1000

X7 = 0x208

**b.** For the register values shown above, what is the value of x7 for the following sequence of instructions?

slli x7, x6, 4

x7 = 0x2345678123456780

**c.** For the register values shown above, what is the value of x7 for the following sequence of instructions?

srli x7, x5, 3

X7 = 0x15555555

andi x7, x7, 0xFEF

X7 = 0x15555545

**Problem 3:**

For each RISC-V instruction below, identify the instruction format and show, wherever applicable, the value of the opcode (op), source register (rs1), source register (rs2), destination register (rd), immediate (imm), func3, func7 fields. Also provide the 8 hex char (or 32 bit) instruction for each of the instructions below

add x5, x6, x7

addi x8, x5, 512

ld x3, 128(x27)

sd x3, 256(x28)

beq x5, x6 ELSE #ELSE is the label of an instruction 16 bytes larger #than the current content of PC

**add** x3, x0, x0

auipc x3, FFEFA

jal x3 ELSE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Type | Opcode,func7,func3 | rs1 | rs2 | rd | imm |
| add x5, x6, x7 | R | 0x33,0000000,000 | 6 | 7 | 5 | - |
| addi x8, x5, 512 | I | 0x13,-,000 | 5 | - | 8 | 512 |
| ld x3, 128(x27) | I | 0x03,-,011 | 27 | - | 3 | 128 |
| sd x3, 256(x28) | S | 0x23,-,011 | 28 | 3 | - | 256 |
| beq x5, x6 ELSE | SB | 0x63,-,000 | 5 | 6 | - | 16 |
| add x3, x0, x0 | R | 0x33,0000000,000 | 0 | 0 | 3 | - |
| auipc x3, FFEFA | U | 0x17,-,- | - | - | 3 | FFEFA |
| jal x3 ELSE | UJ | 0x6F,-,- | - | - | 3 | 16 |

**Problem 4:**

**(a)** For the following C statement, write a minimal sequence of RISC-V assembly instructions that performs the identical operation. Assume x5 = A, and x11 is the base address of C.

A = C[0] << 16;

lw x5, 0(x11)

slli x5, x5, 16

**(b)** Find the shortest sequence of RISC-V instructions that extracts bits 12 down to 7 from register x3 and uses the value of this field to replace bits 28 down to 23 in register x4 without changing the other bits of registers x3 or x4. (Be sure to test your code using x3 = 0 and x4 = 0xffffffffffffffff. Doing so may reveal a common oversight.)

addi x8, x0, 0x3f

slli x8, x8, 7

and x9, x3, x7

slli x8, x8, 16

xori x8, x8, -1

and x4, x4, x8

slli x9, x9, 16

or x4, x4, x9

**(c)** Provide a minimal set of RISC-V instructions that may be used to implement the following pseudoinstruction:

not x5, x6 // bit-wise invert

*[Hint: note that there is no ‘not’ instruction in RISCV. However, an XOR immediate instruction could be used]*

Applying the xor function between any binary string and a string of 1’s reverses the binary string. Thus, we can use -1 using 2s complement obtained by reversing each bit in 1 and adding 1 to it, which is 1111 1111 1111 1111 to get the bit-wise invert:

XORI x5, x6 -1

**Problem 5:**

Suppose the program counter (PC) is set to 0x60000000hex.

1. 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 jump distance is limited, and the maximum distance is determined by the length of imm in jal. The length of imm in jal is 20 bits, therefore the number of bits that can be shifted is 2^(20-1(Sign bit)+1(Ellipsis bit)).

The upper boundary of the range is :

0x60000000 + (2^20 - 2):

0110 0000 0000 0000 0000 0000 0000 0000 +

0000 0000 0000 1111 1111 1111 1111 1110

---------------------------------------------------

0110 0000 0000 1111 1111 1111 1111 1110

i.e. 0x600FFFFE

The lower boundary of the range is:

0x60000000 - 2^20

0110 0000 0000 0000 0000 0000 0000 0000 +

1111 1111 1111 0000 0000 0000 0000 0000

-------------------------------------------------------

0101 1111 1111 0000 0000 0000 0000 0000

i.e. 0x 5FF00000

**b.** 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?)

For the same reason, and the length of imm of beq is 12. Therefor The upper boundary of the range is :

0110 0000 0000 0000 0000 0000 0000 0000 +

0000 0000 0000 0000 0000 1111 1111 1110

---------------------------------------------------

0110 0000 0000 0000 0000 1111 1111 1110

i.e. 0x60000FFE

The lower boundary of the range is:

0110 0000 0000 0000 0000 0000 0000 0000 +

1111 1111 1111 1111 1111 0000 0000 0000

-------------------------------------------------------

0101 1111 1111 1111 1111 0000 0000 0000

i.e. 0x5FFFF000.

**Problem 6:**

Assume that the register x6 is initialized to the value 10. What is the final value in register x5 assuming the x5 is initially zero?

LOOP: beq x6, x0, DONE

addi x6, x6, -1

addi x5, x5, 2

jal x0, LOOP

DONE:

1. For the loop above, write the equivalent C code. Assume that the registers x5 and x6 are integers acc and i, respectively.

acc = 0;

i = 10;

while (i != 0){

i = i -1;

acc = acc + 2;

}

1. For the loop written in RISC-V assembly above, assume that the register x6 is initialized to the value N. How many RISC-V instructions are executed?

For there are four instructions each loop, and there are N loops if x6 = N, there will be 4N instructions after the loop is end. Then there is one more instruction: “DONE”.

Therefore the total number of instructions will be 4N + 1.

1. For the loop written in RISC-V assembly above, replace the instruction “beq x6, x0, DONE” with the instruction “blt x6, x0, DONE” and write the equivalent C code.

LOOP: blt x6, x0, DONE

addi x6, x6, -1

addi x5, x5, 2

jal x0, LOOP

DONE:

acc = 0;

i = 10;

while (i >= 0){

i = i -1;

acc = acc + 2;

}

**Problem 7:**

**a.** Translate the following C code to RISC-V assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers x5, x6, x7, and x29, respectively. Also, assume that register x10 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;

addi x7, x0, 0

LOOPI:

bge x7, x5, ENDI

addi x30, x10, 0

addi x29, x0, 0

LOOPJ:

bge x29, x6, ENDJ

add x31, x7, x29

sd x31, 0(x30)

addi x30, x30, 16

addi x29, x29, 1

jal x0, LOOPJ

ENDJ:

addi x7, x7, 1

jal x0, LOOPI

ENDI:

**b**. How many RISC-V instructions does it take to implement the C code from 7a. above? If the variables **a** and **b** are initialized to **10** and **1** and all elements of **D** are initially 0, what is the total number of RISC-V instructions executed to complete the loop?

To implement the C code from &a, there are 12 instructions.

If the a and b are initialized to **10** and **1**, the total number of instructions is 1 + (3\*10+1) + (6\*1+1)\*10 + 2\*10 = 122.

**Problem 8:**

Consider the following code:

lb x6, 0(x7)

sd x6, 8(x7)

Assume that the register x7 contains the address **0×10000000** and the data at address is 0×**11**223344556677**88**.

1. What value is stored in 0×10000007 on a bigendian machine?

0x88 is stored in 0×10000007 on a bigendian machine

1. What value is stored in **0×10000007** on a littleendian machine?

0x11 is stored in 0×10000007 on a littleendian machine

**Problem 9:**

Write the RISC-V assembly code that creates the 64-bit constant 0x1234567812345678hex and stores that value to register x10.

lui x10, 0x12345 //use lui to input the first 5 number(20 bits)

addi x10, x10, 0x678 //add next 3 number(12 bits)

slli x10, x10, 32 // move the 8 number to the upper half

lui x5, 0x12345//input next 5 number (20 bits)

addi x5, x5, 0x678 //add next 3 number (12 bits)

add x10, x10, x5 //add up the two parts

**Problem 10:** Assume that **x5** holds the value **12810.**

1. For the instruction **add x30, x5, x6,** what is the range(s) of values for **x6** that would result in overflow?

There is an overflow if 128 + x6 > 2^31 − 1 In other words, if x6 > 2^31 − 129

There is also an overflow if 128 + x6 < −2^31 that is, if x6 < −2^31 − 128 (which is impossible given the range of x6 )

**b.** For the instruction **sub x30, x5, x6,** what is the range(s) of values for **x6** that would result in overflow?

There is an overflow if 128 – x6 > 2^31 − 1. In other words, if x6 < −2^31 + 129

There is also an overflow if 128 – x6 < −2^31. In other words, if x6 > 2^31 + 128 (which is impossible given the range of x6 )

**c.** For the instruction **sub x30, x6, x5,** what is the range(s) of values for **x6** that would result in overflow?

There is an overflow if x6 − 128 > 2^31 − 1. In other words, if x6 < 2^31 + 127 (which is impossible given the range of x6 )

There is also an overflow if x6 − 128 < −2^31 In other words, if x6 < −2^31 + 128