EECS 112 & CSE 132, FALL 2017

Homework 2

Due date: October, 26, 2017

Student ID:	7	2	7	9	7	0	3	8	Name: Michael Chai

1) Write the RISC-V assembly code for the following C statement. Assume that elements of arrays A and B are 32 bits. Also, what is the hexadecimal machine code for those instructions?

```
B[2*k] = A[i - j]
```

Hint: *Assume that variables i, j and k are assigned to registers X_{10}, X_{11} and X_{12}. Also, the base addresses of arrays A and B are registers X_5 and X_6. Suppose that register X_{30} is used as the temporary register.*

note: x31 is secondary temporary register

sub x30, x10, x11	01000 00 01011 01010 000 11110 0110011
slli x30, x30, 2	00000 0000010 11110 001 11110 0010011
add x30, x5, x30	00000 00 11110 00101 000 11110 0110011
lw x30, 0(x30)	000000 11110 11110 010 00000 0100011
slli x31, x12, 3	00000 0000011 01100 001 11111 0010011
add x31, x6, x31	00000 00 11111 00110 000 11111 0110011
sw x30, 0(x31)	000000 11111 11110 010 00000 0000011

2) Translate the following RISC-V code to C. Assume that the variable f is assigned to register X_5. Also, the base address of array A is in register X_{10}. X_{30} and X_{31} are the temporary registers.

```
addi    x30,  x10,  8
```
x30 = &A + 8·1 = &A[1]

```
addi    x31,  x10,  0
```
x31 = &A + 8·0 = &A[0]

```
sd      x31,  0(x30)
```
A[1] = &A[0]

```
ld      x30,  0(x30)
```
x30 = A[1]

```
add     x5,   x30,  x31
```
f = x30 + x31 → f = A[1] + &A[0]

A[1] = &A[0]

f = A[1] + &A[0]

3) Implement the following RISC-V pseudo instructions using the recommended instructions.

Pseudo Instruction	**Meaning**	**Recommended Instruction**
`not rd, rs`	`rd = ~rs`	`xori` rd, rs, -1
`nop`	`Neither the registers nor the memory contents change`	`addi` x0, x0, 0
`mov rd, rs`	`rd = rs`	`addi` rd, rs, 0
`j label1`	`Jump to a label (not linking)`	`jal` x0, label1

	not
0	1
1	0

$\sim A = A \oplus -1$

↓

111…111

		xor
0	0	0
1	0	1
0	1	1
1	1	0

4) Translate the following C code to RISC-V assembly code. Assume that the values of a, b, i and j are in registers X_5, X_6, X_7 and X_{29}, respectively. Also, assume that register X_{10} holds the base address of the array D.

```
for ( i = 0 ; i < a ; i++ )
    for( j = 0 ; j <b ; j++)
        D[4*j] = 3*i;
```

```
          li    x7, 0               // i = 0
loop1:    bge   x7, x5, endloop1    // i ≥ a ?, if so, then end outer loop
loop2:    li    x29, 0              // j = 0
          bge   x29, x6, endloop2   // j ≥ b ?, if so, then end inner loop
          add   x30, x7, x7         // temp1 = i + i = 2i
          add   x30, x30, x7        // temp1 = temp1 + i = 3i
          slli  x31, x29, 5         // temp2 = j<<5 = (4·j)·8 = (2²·j)·2³
          add   x31, x10, x31       // temp2 = baseD + (4·j)·8 = base D[4·j]
          sd    x30, 0(x31)         // D[x31 + 0] = x30 → D[4·j] = 3·i
          addi  x29, x29, 1         // j++
          j     loop2               // then repeat loop
endloop2: addi  x7, x7, 1           // i++
          j     loop1               // then repeat loop
endloop1: ~
```

loop2:

5) Fill out the following table:

	Smallest absolute value (excluding 0)	Largest absolute value (excluding infinity)
IEEE 754 single precision	S=X E(8)=-126 M(23)=0 $\pm 1.0 \cdot 2^{-126}$	S=X E(8)=127 M(23)=1111···1111 $\pm 10.0 \cdot 2^{127}$
IEEE 754 double precision	S=X E(11)=-1022 M(52)=0 $\pm 1.0 \cdot 2^{-1022}$	S=X E(8)=1023 M(52)=1111···1111 $\pm 10.0 \cdot 2^{1023}$

$\pm 1.2 \cdot 10^{-38}$	$\pm 3.4 \cdot 10^{38}$
$\pm 2.2 \cdot 10^{-308}$	$\pm 1.8 \cdot 10^{308}$

6) Convert the following numbers to IEEE 754 single precision binary representation:

a. -115~~.~~964116992. x ~~10^9~~

$1\,1011_{32}\,0000 \cdots 0000$

↑
36

S=1

E=36+127=163

1010 0011

M=1011 0000 ····

1	1010 0010	101 1000 0000 0000 0000 0000

b. .25 x 2^{-130}

$(1 \cdot 2^{-2}) \cdot 2^{-130} = 1 \cdot 2^{-132}$ 1000 →

S=0

E=-132+127=-5 ← must be ≥1

M=0

Number cannot be store in single precision floating point

DOUBLE PRECISION

S=0

$E=-132+1023=891=(1101111011)_2$ →

M=0

(52 bits wide)

0	1101111011	0000····0000

7) To add two floating point numbers a and b with equal exponents, we can simply add the mantissa together, and then adjust the exponent by 1 if there is a carry. When exponents are not equal, we should modify the mantissa of one of the two numbers a and b (by shifting it) such that the exponents become equal, and then we can add the mantissa together. In doing so, <u>we should not add any significant zeros to either a or b</u> (See 'significant figure' on Wikipedia if you don't know what it is. You may also want to consult Google on 'floating point addition').

Consider a variation of IEEE 754 with 4 bits for exponent, 8 bits for fraction, and a bias of 7. Add the following numbers, once assuming to have unlimited precisions, and once under this variation of IEEE 754. Calculate the numerical error between the two cases.

$A = (1.01100000)_2 \times 2^{-3}$, $B = (1.01100001)_2 \times 2^{-4}$

A = 0.001011000000
B = 0.000101100001

0.0100,0010,0001

A: S = 0
E = −3+7 = 4 = 0100
M = 01100000

B: S = 0
E = −4+7 = 3 = 0011 → E = 0100
M = 01100001

S = 0
M = 10110000

S = 0 E = 0100 → E = 0101 = 5
M = 00010000
↳1

exponent = 5 − 7 = −2

$(1.0001) \cdot 2^{-2}$
0.0100,01

difference = 0.0000 0000 0001 $= 2^{-12} = 0.000244$

absolute error $= 2^{-12} = 2.44 \cdot 10^{-4}$

relative error $= 2.44 \cdot 10^{-4} / 2.58 \cdot 10^{-1} = 9.46 \cdot 10^{-4} = 0.0946\%$

Good Luck