NYU Tandon School of Engineering

Fall 2023, ECE 6913 INET Quiz 1

---

1. *Write down the RISC V code for the following tasks:*

Assume Base address of arrays `A, B, C` are in registers `x5, x6, x7`

Assume variables `f, g, i` are in registers `x8, x9, x10`

Implement in RISC V these line of code in C:

(i) `f = g - A[B[C[64]]]`

```
lw x7, 256(x7) # C[64]
slli x7, x7, 2
add x6, x6, x7 # &B[C[64]]
lw x6, 0(x6) # B[C[64]]
slli x6, x6, 2
add x5, x5, x6 # &A[B[C[64]]]
lw x5, 0(x5) # A[B[C[64]]]
sub x8, x9, x5 # f = g - A[B[C[64]]]
```

(ii) `f = g - A[C[16] + B[32]]`

```
lw x6, 128(x6) # B[32]
lw x7, 64(x7) # C[16]
add x6, x6, x7 # B[32] + C[16]
slli x6, x6, 2
add x5, x5, x6 # &A[B[32] + C[16]]
lw x5, 0(x5) # A[B[32] + C[16]]
sub x8, x9, x5 # f = g - A[B[32] + C[16]]
```

(iii) `A[i] = 4B[8i-81] + 4C[32i+32]`

```
slli x11, x10, 5 # 32i
addi x11, x11, 32 # 32i+32
slli x11, x11, 2
add x7, x7, x11 # &C[32i+32]
lw x7, 0(x7) # C[32i+32]
slli x7, x7, 2 # 4C[32i+32]
slli x12, x10, 3 # 8i
addi x12, x12, -81 # 8i-81
slli x12, x12, 2
add x6, x6, x12 # &B[8i-81]
lw x6, 0(x6) # B[8i-81]
slli x6, x6, 2 # 4B[8i-81]
add x6, x6, x7 # 4C[32i+32] + 4B[8i-81]
slli x13, x10, 2
add x5, x5, x13 # &A[i]
sw x6, 0(x5) # A[i] = 4B[8i-81] + 4C[32i+32]
```

2. Write a sequence of RISC V instructions that swap `byte 4 (bits 31:24)` with `byte 1 (bits 7:0)` in a 4 byte (32b) long word

```
Assume the 4 byte word is stored in x5
andi x6, x5, 0xFF000000 # extract byte 4
srli x6, x6, 24     # shift extracted byte 4 to bits 7:0
andi x7, x5, 0x000000FF # extract byte 1
slli x7, x7, 24 # shift extracted byte 2 to bits 31:24
or x8, x6, x7 # combine swapped byte 1 and 4
andi x5, x5, 0x00FFFF00 # Zero out bits 31:24 and 7:0 in original register
```

```
or x5, x5, x8 # Load swapped byte 1 and 4 in original register
```

3. One possible performance enhancement is to do a shift and add instead of an actual multiplication. Since 9×6, for example, can be written $(2\times2\times2+1\times6$, we can calculate 9×6 by shifting 6 to the left three times and then adding 6 to that result. Show the best way to calculate $0x3A_{hex} \times 0x5F_{hex}$ using shifts and adds/subtracts. Assume both inputs are 8-bit unsigned integers.

$0x3A_{16}$ = 0011 1010 = 58_{10}

$0x5F_{16}$ = 0101 1111 = 95_{10}

58 x 95 = 5510_{10} = $0x1586_{16}$

$0x3A_{16}$ = 00111010_2 = 58_{10}, and 58 = 32+16+8+2 = $2^5 + 2^4 + 2^3 + 2^1$

So,

We can shift 0x5F left 5 places = 0 1011 1110 0000 = 3040_{10} = $0xBE0_{16}$

then add 0x5F left shifted 4 places = 0101 1111 0000 = 1520_{10} = $0x5F0_{16}$

then add 0x5F left shifted 3 places = 0010 1111 1000 = 760_{10} = $0x2F8_{16}$

then add 0x5F left shifted 1 place = 1011 1110 = 190_{10} = $0xBE_{16}$

Adding 3040+1520+760+190 = 5510 = $0x1586_{16}$.

So, we can use 4 shifts and 3 adds instead of actual multiplication.

4. Assume a 10-bit floating point representation format where the Exponent Field has 4 bits and the Fraction Field has 5 bits and the sign bit field uses 1 bit

S	Exponent Field: 4 bits	Fraction Field: 5 bits

a. What is the representation of **-8.80158×10^{-2}** in this Format
[assume:

bias = 2^{N-1} - 1= 2^{4-1} -1= 7 (where N = number of exponent field bits) for normalized representation, and 1-bias = **-6 = bias for denormalized representation**]. What 10-bit pattern represents the number **−0.125 = −1/8?** What base-10 integer or fraction does this 10-bit floating point representation format of **0101001001** equal to?

i) Representation of -8.80158 x 10-2 in this Format

-8.80158 x 10-2 = -0.0880158

Fraction = 00010110100010000001101 = 1.0110100010000001101 x 2-4 (Normalized)

Value of Exponent = -4 = Exponent – Bias = Exponent - 7

Exponent = 3 (or 0011 in binary)

Sign = 1

Binary Representation

1 0011 01101

ii) 10-bit representation of -0.125

Fraction = 001 = 1.000 x 2-3

Value of Exponent = -3 = Exponent – Bias = Exponent - 7

Exponent = 4 (or 0100 in binary)

10-bit representation

1 0100 00000

iii) Equivalent base 10 integer or fraction of 0 1010 01001

In 10-bit representation, MSB is sign bit (S), next 4 bit is Exponent and 5 left bits are Fraction

S = 0 E = 1010 F = 01001

Decimal of E = 10

Value of Exponent = Exponent – Bias = 10 -7 = 3

Value of a Normalized Floating-Point Number is

$$(-1)^S X\ (1.F) X\ 2^{(E-Bias)}$$

Value in decimal = (1.01001) x 23 = (1010.01)2 = 10.25

b. What is the range of representation for *positive* *normalized* numbers – what is the largest and smallest *normalized* number represented in this format? What is the range of representation for *positive* *denormalized* numbers? – what is the largest and smallest *denormalized* number represented in this format?

RANGE OF REPRESENTATION FOR POSITIVE NORMALIZED NUMBER

Largest Positive normalized number

For given 10-bit representation, range of Exponent is 1 to 14

For the largest positive normalized number, Exponent will be 14 and Fraction will have all bit set to 1 $(11111)_2$.

Value of exponent = E - 7 = 14 - 7 = 7

S = 0 Val(E) = 7 Fraction = 11111

Value of Representation = $(1.11111)_2$ x 2^7

= $(2 - 2^{-5})$ x 2^7

= 2^8 x $(1 - 2^{-6})$

Or 127 x 2^3

Smallest positive normalized number

For smallest positive normalized number, Exponent will be 1 and Fraction will have all bit set to 0 $(00000)_2$.

Value of Exponent = E - 7 = 1 - 7 = -6

S = 0 Val(E) = -6 Fraction = 00000

Value of Representation = $(1.000000)_2$ x 2^{-6}

Therefore, range of positive normalized number is 2^{-6} to 2^{8} x $(1 - 2^{-6})$

RANGE OF REPRESENTATION FOR POSITIVE DENORMALIZED NUMBERS

Largest Positive denormalized number

For the largest positive denormalized number, Exponent will be 0 and Fraction will have all bit set to 1 $(11111)_2$.

Value of Exponent = 1 – Bias = -6

S = 0 Val(E) = -6 Fraction = 11111

Value of Representation = $(0.11111)_2 \times 2^{-6} = (1 - 2^{-5}) \times 2^{-6}$

Smallest positive denormalized number

For the smallest positive denormalized number, Exponent will be 0 and Fraction will have all bit set to 0 except the least significant $(00001)_2$.

Value of Exponent = 1 - Bias = -6

S = 0 Val(E) = -6 Fraction = 00001

Value of Representation = $(0.00001)_2 \times 2^{-6} = 2^{-11}$

Therefore, range of positive normalized number is 2^{-11} to $(1 - 2^{-5}) \times 2^{-6}$

5. Using the IEEE 754 (Single Precision) floating-point format, write down the bit pattern that would represent −1/4, -1/3

In single Precision floating point format, Sign has 1 bit, Exponent is of 8 bits and Fraction takes 23 bits. Bias is 127

i) -1/4 = -0.25

Fraction = (01000000000000000000000)2 = (1.00000000000000000000) x 2^{-2}
Normalized form

Value of Exponent = -2

Value of Exponent = -2 = Exponent – Bias = Exponent – 127

Exponent = 125 (or 01111101)

Sign = 1 (for negative number)

Binary representation

1 01111101 00000000000000000000000

ii) -1/3 = - 0.33333333333

Fraction = (01010101010101010101010)2 = (1.01010101010101010101010) x 2^{-2}

Value of exponent = -2

Value of Exponent = -2 = Exponent – bias = Exponent – 127

Exponent = 125 (or 01111101)

Sign = 1 (for negative number)

Binary representation

1 01111101 01010101010101010101010

6. Your company has just bought a new Intel Core i5 dual core processor, and you have been tasked with optimizing your software for this processor. You will run two applications on this dual core, but the resource requirements are not equal. The first application requires 70% of the resources, and the other only 30% of the resources. Assume that when you parallelize a portion of the program, the speedup for that portion is 2.

a. Given that 40% of the first application is parallelizable, how much speedup would you achieve with that application if run in isolation?

Since we are running the first application in isolation,

$$\text{Speedup} = \frac{1}{(1-0.4)+(\frac{0.4}{2})} = 1/(0.6 + 0.2) = 1/0.8 = 1.25$$

Therefore, speedup is 1.25

b. Given that 99% of the second application is parallelizable, how much speedup would this application observe if run in isolation?

Since we are running the second application in isolation,

$$\text{Speedup} = \frac{1}{(1-0.99)+(\frac{0.99}{2})} = 1/(0.01 + 0.495) = 1/0.505 = 1.98$$

Therefore, speedup is 1.98

c. Given that 40% of the first application is parallelizable, how much overall system speedup would you observe if you parallelized it?

Resources needed for first application = 70%

Percent of first application that can be parallelized = 40%

Hence, percent of overall system that can be parallelized = 0.7 * 0.4 = 28%

$$\text{Speedup} = \frac{1}{(1-0.28)+(\frac{0.28}{2})} = 1/(0.72 + 0.14)= 1/0.86$$

$= 1.1627 \approx 1.16$

Therefore, speedup is 1.16