Michael Lankford

Homework 3

2/14/2023

1.

* An Arithmetic Logic Unit(ALU) is hardware that performs addition, subtraction, and usually logical operations like AND and OR.
* Overflow is a situation where a positive exponent becomes too large to fit in the exponent field. Underflow is a situation where a negative exponent becomes too large to fit in the exponent field. You can detect overflow for unsigned numbers by looking at the carry-out of the leftmost bit. You can also detect overflow for 2’s complement signed numbers if the sum of two positive numbers yields a negative result or if the sum of two negative numbers yields a positive result.
* Exceptions and interrupts are the same thing. They are unscheduled events that disrupt a programs execution and are used to detect overflow by MIPS. They come from outside the processor and the register that is used to contain the address of the instruction is the exception program counter or EPC.
* Saturating Arithmetic is a version of arithmetic where all operations are limited to a fixed range between a minimum and maximum value.
* Integer numbers are whole numbers that can be positive, negative, or zero. Floating point numbers are computer arithmetic that represents numbers in which the binary point is not fixed.
* Scientific notation is a notation that renders numbers with a single digit to the left of the decimal point. Normalized numbers are numbers in floating-point notation that has no leading 0s.
* FP number representations have to compromise between the size of the fraction and the size of the exponent. This is a tradeoff between precision and range, and it comes out to the function (-1)S x F x 2E.

2. Exercise 3.1

5ED4-07A4 = 0101 1110 1101 0100

– 0000 0111 1010 0100

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

0101 0111 0011 0000 which equals 5730 in hexadecimal

3. Exercise 3.3

5ED4 in binary is 0101111011010100. Hexadecimal is an attractive numbering system for representing computer values because it is easier to read compared to 0s and 1s, especially because binary numbers get big extremely fast.

4. Exercise 3.4

4365 – 3412 in binary is 100 011 110 101

- 011 100 001 010

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

000 111 101 011 which equals 0753 in octal.

5. Exercise 3.11

151

+ 214

--------

365 but because it is an 8-bit integer, and we are using saturating arithmetic the max value is 255 so it is 255 and not 365.

6. Exercise 3.14

For hardware, the time necessary to perform a multiply is 96 time-units because the integer is 8 bits wide, the operations take 4 time-units, and there are 3 operation steps when it runs. You multiply each of those values and it comes out to 96 time-units.

For software, the time necessary to perform a multiply is 160 time-units because the integer is 8 bits wide, the operations take 4 time-units, and there are 5 operation steps when it runs. You multiply each of those values and it comes out to 160-time units.

7. Exercise 3.16

The time necessary to perform a multiply is 28 time-units because the operations take 4 time-units and there are log2(32) = 5 levels of adders. The amount of adders on the top level is 7 so you multiply 7 by the 4 time-units and you get a total of 28 time-units.

8. Exercise 3.17

0x33 \* 0x55 = 0011 0011 \* 0101 0101 = (25 \* 24 \* 21 + 1) \* (26 \* 24 \* 22 + 1)

Step 1) Take 0x55 and shift it left 5 bits

Step 2) Add it to the total

Step 3) Take 0x55 and shift it left 4 bits

Step 4) Add it to the total

Step 5) Take 0x55 and shift it left 1 bit

Step 6) Add it to the total

Step 7) Add 0x55 to the total

9. Exercise 3.22

0x0C000000 is converted to 0000 1100 0000 0000 0000 0000 0000 0000 in binary. The IEEE 754 floating point standard says that the 1st bit is the sign bit, the next 8 bits are the exponent bits, and the last bits are the mantissa bits. According to this, the binary number would be separated like this: 0 00011000 00000000000000000000000.

X = (-1)S \* (1 + Fraction) \* 2(exponent – bias) = (-1)0 \* (1 + 0) \* 2(24 – 127) = 1 \* 1 \* 2-103 = 1.0 \* 2-103

10. Exercise 3.23

63 / 2 = 31 (1)

31 / 2 = 15 (1)

15 / 2 = 7 (1)

7 / 2 = 3 (1)

3 / 2 = 1 (1)

1 / 2 = 0 (1)

0.25 \* 2 = .5 (0)

0.50 \* 2 = 1 (1)

0.00 \* 2 = 0 (0)

63.25 is converted to 111111.01 x 20 = 1.1111101 x 25+127 = 1.1111101 x 2132

132 is converted to 10000100 in binary

Filling in the IEEE 754 format, we get 0 10000100 11111010000000000000000

0100 0010 0111 1101 0000 0000 0000 0000 = 0x427D0000