CS147: Computer Architecture
Project #2
Half-Precision Arithmetic

Student ID: _____
Student Name: _____
Points out of 100: _____

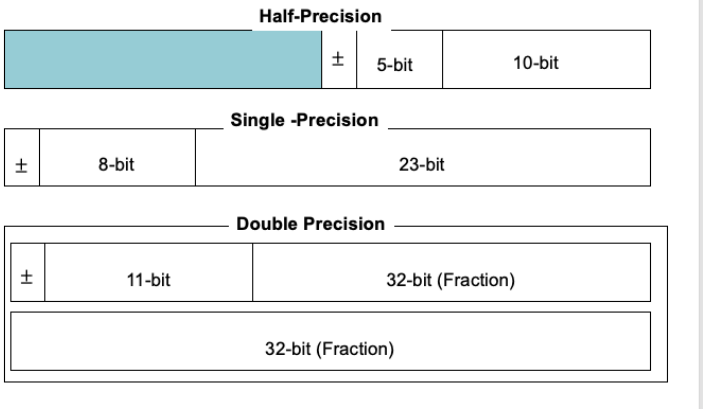
- This document may be updated with more information and/or hints as needed.
- The provided codes may contain bugs and may be updated accordingly.

A. Purpose

The MIPS instructions have instructions for handling single precision and double precision numbers. But there are no instructions implemented for handling the half precision numbers. In this project, we are going to implement these functions to provide the arithmetic operations for half precision numbers. Before working on this project, you better finish HW3 first to get you familiar with the IEEE encoding for floating point numbers and the associated instructions (more details in the text book or on web pages like here or here.)

B. IEEE 754 Standards

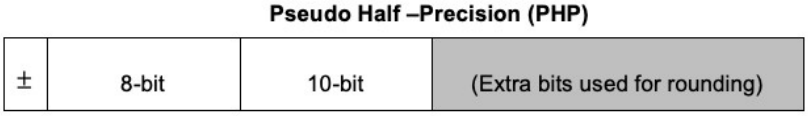
IEEE 754 defines three standards for half-, single-, and double-precision floating point numbers as shown below.



To facilitate the work of this project, we purposely introduce another so-called pseudo half precision format (see below). The coding of this project can be simplified for the following reasons.

- Single precision instructions can be used for the PHP numbers.
    - PHP is just a special single precision format where some bits are not used.
- The intermediate results have 23 bits in fraction that can be used for rounding to produce the PHP that has only 10 bits in fraction.
- Moreover, we can print out the decimal values of PHP numbers just like the single precision numbers with the same syscall code.

```
li          $v0,  2        # print single or half-precision numbers saved in $f12
syscall
```



B.1 Special Numbers

Below are the number presentations in IEEE 754 format, where the maximum exponent for both single precision and PHP are the same. Besides, the first 10 bits in the fraction of PHP is non-zero.

IEEE 754 Symbols

Exponent	Fraction	Object
0	0	± 0
0	≠0	± Denorm
1 to MAX - 1	anything	± Fl. Pt. Num.
MAX	0	±∞
MAX	≠0	NaN

S.P. MAX = 255, D.P. MAX = 2047

P.H.P. MAX = S.P.MAX = 255
H.P.MAX = 32

The arithmetic results of the special numbers should be the consistent in all precision formats. For example,

- Largest numbers + 1 = Infinity + Infinity = Infinity × Infinity = Infinity
- Infinity – Infinity = Infinity / Infinity = NaN
- Infinity ± 1 = Infinity ± 2 = … = Infinity
- 1÷0 = Infinity

Some important numbers in the half precision are:

- Normal numbers
    - The largest numbers in PHP is 65504_{10}.

- The smallest numbers in PHP is -65504_{10}.
- The smallest positive **normal** numbers is 0.00006103515_{10} (2^{-14}).
- The largest negative **normal** numbers is -0.00006103515_{10} (2^{-14})

- Denormal numbers
  - The largest denorm numbers in PHP is (2^{-15}).
  - The smallest denorm numbers in PHP is (-2^{-15}).
  - The smallest positive denorm numbers in PHP is (2^{-24}).
  - The largest negative denorm numbers in PHP is (-2^{-24}).

Note that all denorm numbers in half-precision can be represented by the PHP or single-precision numbers. Therefore, there is no demorm numbers in PHP.

C. Functions to Implement

MIPS has the following instructions for the conversion among different precisions.

- FP ↔ integer conversion
  - cvt.w.s fdst, fsrc | Convert to integer from single
  - cvt.w.d fdst, fsrc | Convert to integer from double
  - cvt.s.w fdst, fsrc | Convert to single from integer
  - cvt.d.w fdst, fsrc | Convert to double from integer
- Single ↔ Double conversion
  - cvt.d.s fdst, fsrc | convert to double from single
  - cvt.s.d fdst, fsrc | convert to single from double

Similarly, we are going to implement the functions that convert numbers among single precision, pseudo and true half precision. In particular, the conversion has to work for special numbers (Infinity, NaN, denormal, largest, smallest)

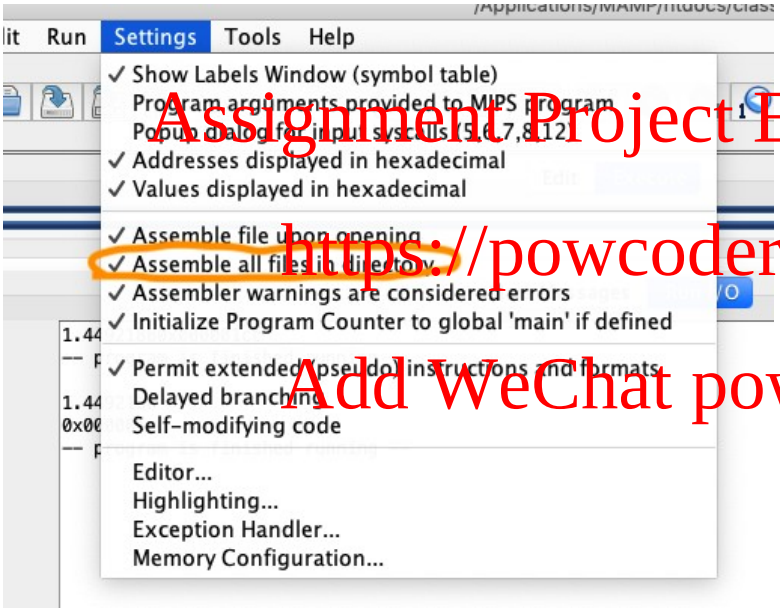
- Conversion between single precision numbers (float) and PHP.
  - cvt.php.s
    - Input: single precision number (in $f12)
    - Output: PHP number (in $f0)
    - Convert to PHP from single
  - Note: No need to implement the other function cvt.s.php since PHP numbers are just a subset of single precision numbers
- Conversion between half-precision and PHP
  - cvt.h.php:
    - Input: a PHP number (in $f12)
    - Output: a (true) half precision (in $v0)
    - Convert to half precision from PHP
  - cvt.php.h:
    - Input: a half precision (in $a0)
    - Output: a PHP number (in $f0)

- Convert to PHP from half precision

MIPS has the following instructions for the arithmetic operation with single or double precisions: add.s, sub.s, mul.s, div.s and sdd.d, sub.d, mul.s and div.d. In this project, we are going to add the following functions to perform the arithmetic operation with half precisions.

- Half Precision Arithmetic:
  - add.php, sub.php, mul.php, div.php
    - Input: Single precision numbers A, B in $f12, $f13
    - Output: a PHP number C = A op B in $f0, where op = +, −, ×, ∨ ÷.

You are required to implement each function in its own file, and all the files of this project have to be placed in the same directory. To assemble all the programs of the project on MARS, you need to turn on the option "To assemble all the files in the same directory".



D. File Description

- Conversion to PHP from single precisions numbers
  - cvt.php.s.asm
- Conversion between PHP and (true) half precision numbers
  - cvt.h.php.asm
  - cvt.php.h.asm
- Arithmetic Operations for PHP numbers
  - add.php.asm
  - sub.php.asm
  - mul.php.asm
  - div.php.asm
- Testing programs

There are two kind of testing programs:

- The first kind are designed to produce the answer keys for some exercises in the textbook.
- The second kind of programs, named test.*.asm, are used to test your programs for the special numbers like infinity, NaN and denorm.
- Ex.3.27.asm
  - For the given number A in Exercise 3.27
    - Output decimal value of A in half precision format
    - Output the encoding of A in half precision format
- Ex.3.30.asm
  - For the given numbers A and B in Exercise 3.30
    - Output decimal value of C=A*B using half precision format
    - Output the encoding of C in half precision format
- Ex.3.31.asm
  - For the same numbers A and B given in Exercise 3.30
    - Output the decimal value of C=A/B using half precision format
    - Output the encoding of C in half precision format
- Ex.3.32.asm
  - For the numbers A, B and C given in Exercise 3.32
    - Output the decimal value of (A+B)+C using half precision format
    - Output the encoding of the sum in half precision format
- Ex.3.33.asm
  - For the same numbers A, B and C given in Exercise 3.32
    - Output the decimal value of A+(B+C) using half precision format
    - Output the encoding of the sum in half precision format
- Ex.3.35.asm
  - For the numbers A, B and C given in Exercise 3.35
    - Output the decimal value of (A*B)*C using half precision format
    - Output the encoding of the product in half precision format
- Ex.3.36.asm
  - For the numbers A, B and C given in Exercise 3.36
    - Output the decimal value of A*(B*C) using half precision format
    - Output the encoding of the product in half precision format
- Ex.3.38.asm
  - For the numbers A, B and C given in Exercise 3.38
    - Output decimal value of X=Ax(B+C) using half precision format
    - Output the encoding of X in half precision format
- Ex.3.39.asm
  - For the same numbers A, B and C given in Exercise 3.38
    - Output the decimal value of Y=(AxB) +(AxC) using half precision format
    - Output the encoding of Y in half precision format

E. Submission:

The following files are provided for the project.

- add.php.asm
- mul.php.asm
- testing.program.zip

To submit your work, compress the following files as a zipped file with file name project.1.zip.

Do not include the above provided programs and the read.me.doc. There is no report to be turned in. The grading is mainly based on the testing using the provided testing programs. See details below.

- cvt.php.s.asm (Major works needed)
- cvt.h.php.asm (Major works needed)
- sub.php.asm (minimal works needed)
- div.php.asm (minimal works needed)
- cvt.php.h.asm

F. Grading criteria:

- The grading is solely based on the correct results produced your work for each of the following testing programs. Each testing program produce the answer keys for the exercises in the textbook. Each test case counts for 6 points.

  1. Ex.3.27.asm
  2. Ex.3.30.asm
  3. Ex.3.31.asm
  4. Ex.3.32.asm
  5. Ex.3.33.asm
  6. Ex.3.35.asm
  7. Ex.3.36.asm
  8. Ex.3.38.asm
  9. Ex.3.39.asm
  10. test.php.0.underflow.asm
  11. test.php.infty.add.asm
  12. test.php.NaN.div.asm
  13. test.php.infty.div.asm
  14. test.php.NaN.mul.asm
  15. test.php.denorm.pos.asm
  16. test.php.denorm.neg.asm
  17. test.cvt.php.h.asm
- Note that
  - Half credit is given if the program assembles and run without crash per test case.
  - Full credit is given if the program passes the test case

- Zero credits for any kind of cheating, including the fake programs that simply print out the answers per test case.