CS F342 Computer Architecture

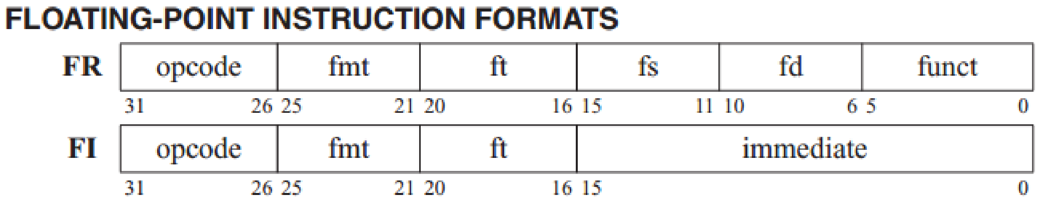
Semester 1: 2019-20

LAB SHEET 3

Goals for the Lab: We build up on Lab Sheet 1 & 2 and explore floating point instructions. We also use some new registers (especially those for floating point operations and overflow / remainder for integer arithmetic – HI / LO) and convert across numeric data types.

Reference for Floating point Instructions – refer to the **MIPS Reference Data Card ("Green Card" -** ARITHMETIC CORE INSTRUCTION SET – Page 1, column 2).

Some of the new instructions (and pseudo instructions) are l.s, mov.s, l.d, mov.d, cvt.d.s, cvt.s.d and similarly single precision and double precision flavours of add, sub, neg, mul, div. For integer overflow we explore mfhi, mhlo etc.



FLOATING-POINT INSTRUCTION FORMATS

FR	opcode	fmt	ft	fs	fd	funct
	31 26	25 21	20 16	15 11	10 6	5 0
FI	opcode	fmt	ft	immediate		
	31 26	25 21	20 16	15 0		

There are 32 single precision floating point registers / 16 double precision values. In QtSpim they are labelled as FG0, FG1, FG2 .. FG31 for single precision and FP0, FP2, FP4 .. FP30 for double precision.

Exercise 1: Scan and Print Float.

```
.data
val1:
      .float 0.001
.text
main:
l.s $f12, val1
li $v0,2
syscall

li $v0, 6
syscall

mov.s $f12, $f0
li $v0,2
syscall

li $v0,10
syscall
```

Exercise 2: Use double precision using l.d and mov.d instead of l.s and mov.s. Also use cvt.d.s and cvt.s.d to convert across precession types.
Non-Evaluative goal: Try to cause and overflow during conversion from single precision to double precision. Look at the value of EPC (exception program counter) cause in the Int Regs[16] tab of QtSpim.

Exercise 3: Add/Sub of float: use instructions like add.d and add.s; sub.s, neg.s etc. Use addition along with negation to implement subtraction.

Exercise 4: Use examples from Lab week 2 for Integer Multiply / Divide. Sample code is given below. Extend it to print the remainder as well.
Hint: use mfhi, mflo and possibly mthi and mtlo for overflow, remainders etc.

```
.data
str0:
      .asciiz "\nMul:"
str1:
      .asciiz "\nDiv:"

w1:
      .word 300
w2:
      .word 7
.text
main:
lw $t0, w1
lw $t1, w2

la $a0, str0
li $v0, 4
syscall

mul $a0, $t0, $t1
li $v0,1   # print from $a0
syscall

la $a0, str1
li $v0, 4
syscall

div $a0, $t0, $t1
li $v0,1   # print from $a0
syscall

# Modify to also print the modulo / remainder


li $v0,10
syscall
```

Exercise 5: Convert Integer to Single precession float and vice versa. Also compute the conversion errors for mathematical operations – e.g. rounding off during divide, precision loss during convert to float and multiple etc. Some of the new commands are cvt.s.w, cvt.w.s, mtc1/mtcz, mfc1/mfcz …

Skeletal algorithm:
//A. Define two integers with relatively large values – but ensure that no overflow in multiplication
//B. multiply and print output – save integer output to a spare register
//C. Convert both integers to single precision float; multiply and print output using float
//D. Convert the output to integer and compare with saved values from step B
//E. Convert both integers to double precision float; multiply and print using double precision float
//F. Convert the output to integer and compare with saved values from step B
//G. Print difference in output of D and E

Non evaluative Goal: Allow for overflow during conversions.

Exercise 6: Explore disassembly for the instructions given below.

1. 44880000
2. 46800060
3. 460208c2
4. 3c041001
5. 44042000
6. 0000000c