

**SI232**  
**Set #11: Fun with Floating Point**  
**(Chapter 3)**

1

2

---

**Exercise #1**

---

(blank space)

- Represent  $+7.25_{10}$  in binary single precision

31	30	29	28	27	26	25	24	23	22	21	20	.	.	.	.	9	8	7	6	5	4	3	2	1	0

3

## Exercise #2

---

(blank space)

- What decimal number is represented by this single precision float?

0100 0001 0110 1000 0000 0000 0000 0000

5

## Exercise #3

---

(blank space)

- Represent  $-17.5_{10}$  in binary single precision and double precision

31	30	29	28	27	26	25	24	23	22	21	20	.	.	.	.	9	8	7	6	5	4	3	2	1	0

31	30	29	28	.	.	.	21	20	19	18	17	.	.	.	9	8	7	6	5	4	3	2	1	0	

31	30	29	28	.	.	.	21	20	19	18	17	.	.	.	9	8	7	6	5	4	3	2	1	0	

7

## MIPS Floating Point Basics

---

- Floating point registers  
\$f0, \$f1, \$f2, ..., \$f31  
Used in pairs for double precision (f0, f1) (f2, f3), ...  
\$f0 not always zero
- Register conventions:
  - Function arguments passed in
  - Function return value stored in
  - Where are addresses (e.g. for arrays) passed?
- Load and store:

```
lwcl $f2, 0($sp)
swc1 $f4, 4($t2)
```

9

## MIPS FP Arithmetic

---

- Addition, subtraction: add.s, add.d, sub.s, sub.d  
`add.s $f1, $f2, $f3`  
`add.d $f2, $f4, $f6`
- Multiplication, division: mul.s, mul.d, div.s, div.d  
`mul.s $f2, $f3, $f4`  
`div.s $f2, $f4, $f6`

10

## MIPS FP Control Flow

---

- Pattern of a comparison: c.\_\_\_.s (or c.\_\_\_.d)  
`c.lt.s $f2, $f3`  
`c.ge.d $f4, $f6`
- Where does the result go?
- Branching:  
`bc1t label10`  
`bc1f label120`

11

## Example #1

---

- Convert the following C code to MIPS:

```
float max (float A, float B) {
    if (A <= B) return A;
    else         return B;
}
```

12

## Example #2

---

- Convert the following C code to MIPS:

```
void setArray (float F[], int index,
              float val) {
    F[index] = val;
}
```

13

## Exercise #1

---

- Convert the following C code to MIPS:

```
float pick (float G[], int index) {
    return G[index];
}
```

14

## Exercise #2

---

- Convert the following C code to MIPS:

```
float max (float A, float B) {
    if (A > B) return A / B;
    else        return B / A;
}
```

(blank space)

15

## Exercise #3

---

(blank space)

- Convert the following C code to MIPS:

```
float sum (float A[], int N) {  
    int j;  
    float sum = 0.0;  
    for (j=0; j<N; j++)  
        sum = sum + A[j];  
    return sum;  
}
```

17

## Exercise #4 – Stretch

---

(blank space)

- Convert the following C code to MIPS:

```
float average (float A[], int N) {  
    int j;  
    float sum = 0.0;  
    for (j=0; j<N; j++)  
        sum = sum + A[j];  
    return sum / N;  
}
```

19

## Chapter Four Summary

---

- Computer arithmetic is constrained by limited precision
- Bit patterns have no inherent meaning but standards do exist
  - two's complement
  - IEEE 754 floating point
- Computer instructions determine “meaning” of the bit patterns
- Performance and accuracy are important so there are many complexities in real machines (i.e., algorithms and implementation).
  
- We are ready to move on (and implement the processor)

21

## Chapter Goals

---

- Introduce 2's complement numbers
  - Addition and subtraction
  - Sketch multiplication, division
- Overview of ALU (arithmetic logic unit)
- Floating point numbers
  - Representation
  - Arithmetic operations
  - MIPS instructions

22