lecture 12

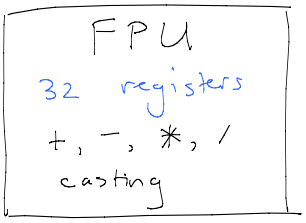
MIPS coprocessors c1 & c0

c1 → floating point

c0 → kernel

Floating Point Unit (FPU)

c1



$f0, $f1, ... $f30, $f31

double precision uses 2 adjacent words

$f0, $f2, $f4, .. $f28, $f30

Floating point operations

single
- add.s $f1, $f0, $f1
- sub.s $f1, $f0, $f1

double
- add.d $f2, $f4, $f6
- sub.d $f2, $f4, $f6

uses $f2, $f3 — uses $f4, $f5 — uses $f6, $f7

NOT ALLOWED

add.s $s0, $s1, $s2

add $f0, $f3, $f7

mul.s $f1, $f0, $f1

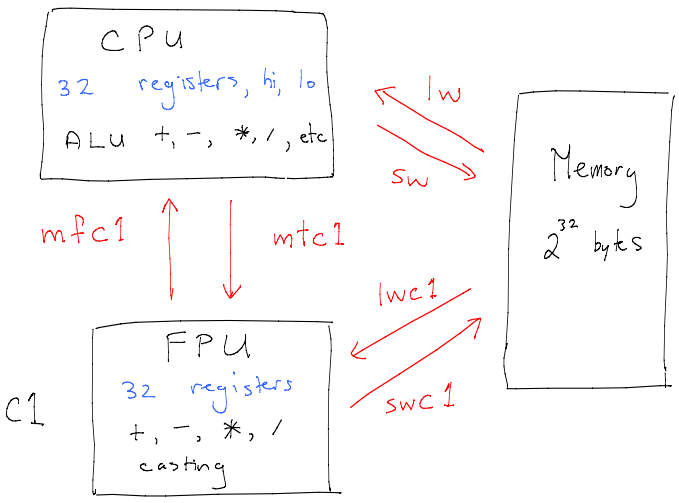
div.s $f1, $f0, $f1

mul.d $f2, $f4, $f6

div.d $f2, $f4, $f6

Note: hi, lo register not used

Data Transfer Operations



CPU ⟷ C1

```
mtc1    $s0, $f0
mfc1    $f0, $s0
```

Note backwards ordering for these instructions.

Loading and storing

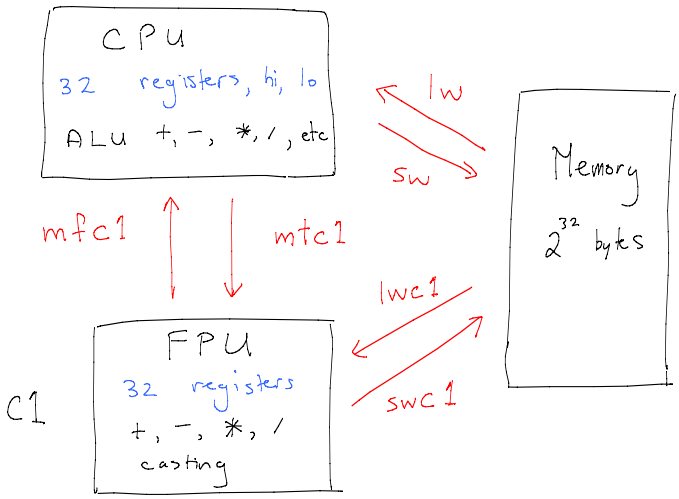
```
lwc1    $f2, 40($s0)
swc1    $f2, 40($s0)
```

↑ address held in CPU register

```
l.d     $f2, 40($s0)
s.d     $f2, 40($s0)
```

pseudo instruction loads two adjacent words

Data Transfer Operations



Casting

```
int     i
float   x
double  y

i = (int) x
x = (float) i
y = (double) x
x = (float) y
```

Casting in MIPS

- performed on C1 (not CPU)

cvt._._

e.g.

```
cvt.s.d     $f3, $f0
cvt.w.s     $f0, $f1
```

↑ int

```
int     i       // $s0
float   x       // $f0
i = (int) x
```

```
mov.s     $f4, $f0
cvt.w.s   $f4, $f4
mfc1      $f4, $s0
```

$f4 is temp # "w" for int

```
int    i          //  $s0
float  x          //  $f0

x = (float) i
```

```
mtc1    $s0, $f0   # note order
cvt.s.w $f0, $f0
```

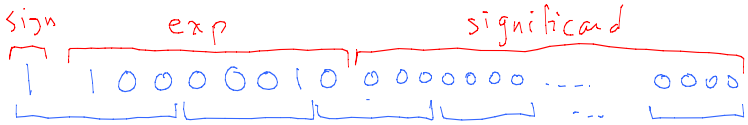
Example

```
addi    $s0, $0, -8
mtc1    $s0, $f0

#   Both  $s0, $f0  have
#     0x  ffffffff8
#   Verify for yourself
```

```
#  $f0 = 0x ffffffff8

cvt.s.w   $f1 , $f0
```




```
#   $f1 = 0xc1000000
```

```
int   i = 841242345
//   2^23 < i < 2^31
int   j = (int)((float) i)
//    i ≠ j
```

```
mtc1      $s0, $f0
cvt.s.w   $f1, $f0
cvt.w.s   $f1, $f1
mfc1      $f1, $s1
#  $s0, $s1  have different values
```

Single precision

1, 8, 23

Double precision

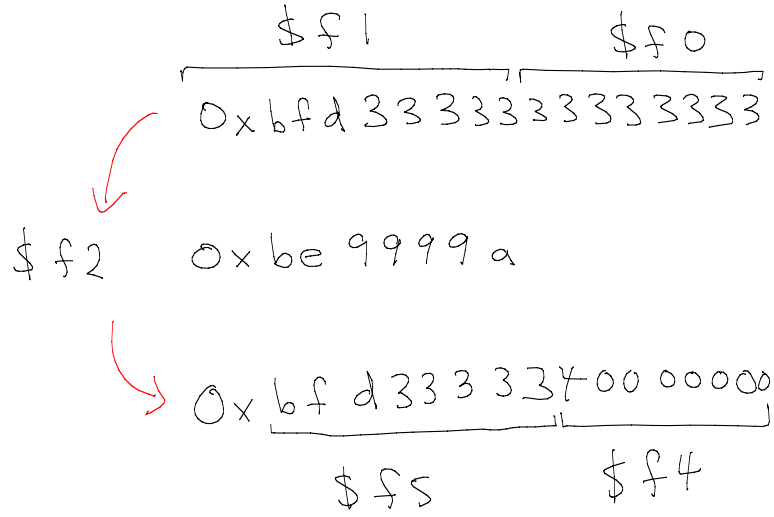
1, 11, 52

(sign, exponent, significand)

```
double   x, y

x =  -0.3
y =  (float) x   // will be cast back to double
```

```
double1: .double  -0.3
l.d      $f0, double1
cvt.s.d  $f2, $f0
cvt.d.s  $f4, $f2
```



other MIPS floating point ops

- trunc, round, floor, ceil
  (convert to integer, but not type "int")

e.g. : trunc.s $f0, $f2
: abs $f0, $f2
: ....

Conditional branch (compare)

c.___.s $f3, $f6
c.___.d $f2, $f6

eq, lt, gt, le, ge

branch if true → bc1t Label
or false → bc1f Label

System Call

		$v0	from/to
print	float	2	$f12
print	double	3	$f12
read	float	6	$f0
read	double	7	$f0

"hardwired" (not just conventions)

lecture 12

MIPS coprocessors c1 & c0

c1: floating point
c0: kernel

Exceptions

- overflow
- division by 0
- bad address
- ....

MIPS Coprocessor 0

registers

- Vaddr – where was bad address
- Status – user? kernel? R/W? interruptable
- Cause – overflow? bad address?
- EPC – return address (after exception resolved)

⋮

Example

⋮	⋮
0x00400070	addiu $s0, $0, 1
0x00400074	sll $s0, $s0, 31
0x00400078	addu $t0, $s0, $s0

⟹ program crashes (overflow error)

	before	after
Vaddr	0	0
Status	0x0000ff11	0x0000ff13
Cause	0	30
EPC	0	0x00400078

Details omitted.

30 means overflow error

```
addi $s0, $0, 8
addi $s1, $0, 0
div  $s0, $s1
```

⟹ Same error (in MARS)

```
str: .asciiz "Hello"

     la  $s0, str
     j   str
```

⟹ MARS assembler error

```
inst: la  $s0, inst
      lw  $t0, 0($s0)
```

	before	after
Vaddr	0	0x00400070
Status	0x0000ff11	0x0000ff13
Cause	0	10
EPC	0	0x00400078

Floating point exceptions ?

- division by zero
- overflow $2^{127} + 2^{127}$
- underflow $2^{-127} \cdot 2^{-24}$
- issues with NaN

⋮

WHAT DOES MARS DO ?

RECALL LECTURE 3

exponent code	exponent value
00000000	reserved
00000001	−126
00000010	−125
⋮	⋮
01111111	0
10000000	1
10000001	2
⋮	⋮
11111110	127
11111111	reserved

RECALL LECTURE 3

Exponent code 11111111

if significand is all 0's

then value is $\pm \infty$ depending on sign bit

else value is "not a number" (NaN)

e.g. variable declared but not yet assigned a value

```
addi    $s0, $0, 1
sll     $s0, $s0, 30
mtc1    $s0, $f0
cvt.s.w $f0, $f0
mul.s   $f0, $f0, $f0
mul.s   $f0, $f0, $f0
mul.s   $f0, $f0, $f0
```

⟹ $f0 = $+\infty$ (no exception)

RECALL LECTURE 3

Exponent code 00000000

"denormalized" numbers

$$\pm 0.\_\_\_\_\_ \times 2^{-126}$$

- belong to $(-2^{-126}, 2^{-126})$
- includes 0

IEEE single precision floating point representation of 0

0 00000000 00000000000000000000000

```
mtc1    $0, $f0
# no need to convert
```

divide by 0

```
positivefloat:  .float  13

l.s     $f0, positive float
mtc1    $0, $f1
div.s   $f2, $f0, $f1
```

⟹ $f2 = +∞

0/0

```
mtc1    $0, $f1
div.s   $f2, $f1, $f1
```

⟹ $f2 = NaN