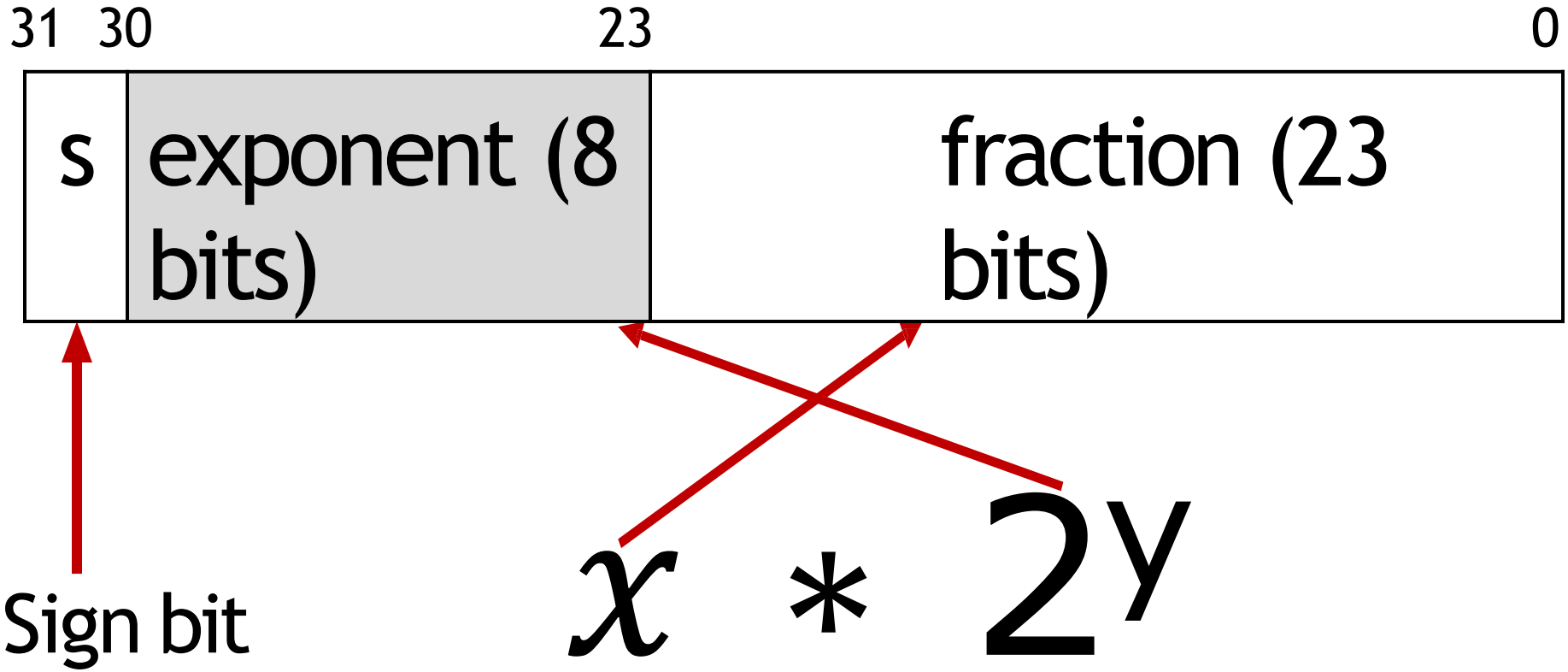
CS 107 Final Review Session

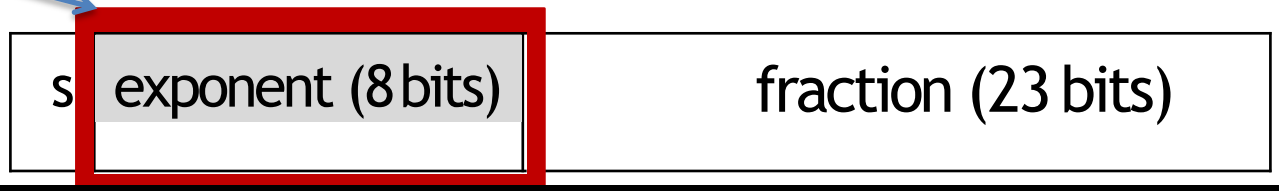
Floats Review

IEEE Single Precision Floating Point



normalized mode:
has bias of 127

Exponent



Exponent (Binary) (Base 10)	Exponent
11111111	RESERVED
11111110	127
11111101	126
11111100	125
…	…
00000011	-124
00000010	-125
00000001	-126
00000000	RESERVED

Pointer Arithmetic Review

```
print_diff(void *a, void *b ) {
    printf (“%zu\n”, (size_t) b – (size_t) a);
}
void *ptr = malloc(16);
char *char_ptr = (char *) ptr;
long *long_ptr = (long *) ptr;
int *int_ptr = (int *) ptr;

print_diff (char_ptr, char_ptr + 1); //1
print_diff (long_ptr, long_ptr + 1);  //8
print_diff (int_ptr, int_ptr + 1);       //4
print_diff (ptr, ptr + 1); //can’t do pointer arith on void *
```

Floats Ex.

Suppose we have an 8-bit representation for a new type of float. 1 sign bit, 3 exponent bits, and 4 fraction bits. Please convert the following to decimal.

1 101 1100

Floats Ex.

Suppose we have an 8-bit representation for a new type of float. 1 sign bit, 3 exponent bits, and 4 fraction bits. The bias is 4. Please convert the following to decimal.

1 101 1100

negative

5 - 4 = 1

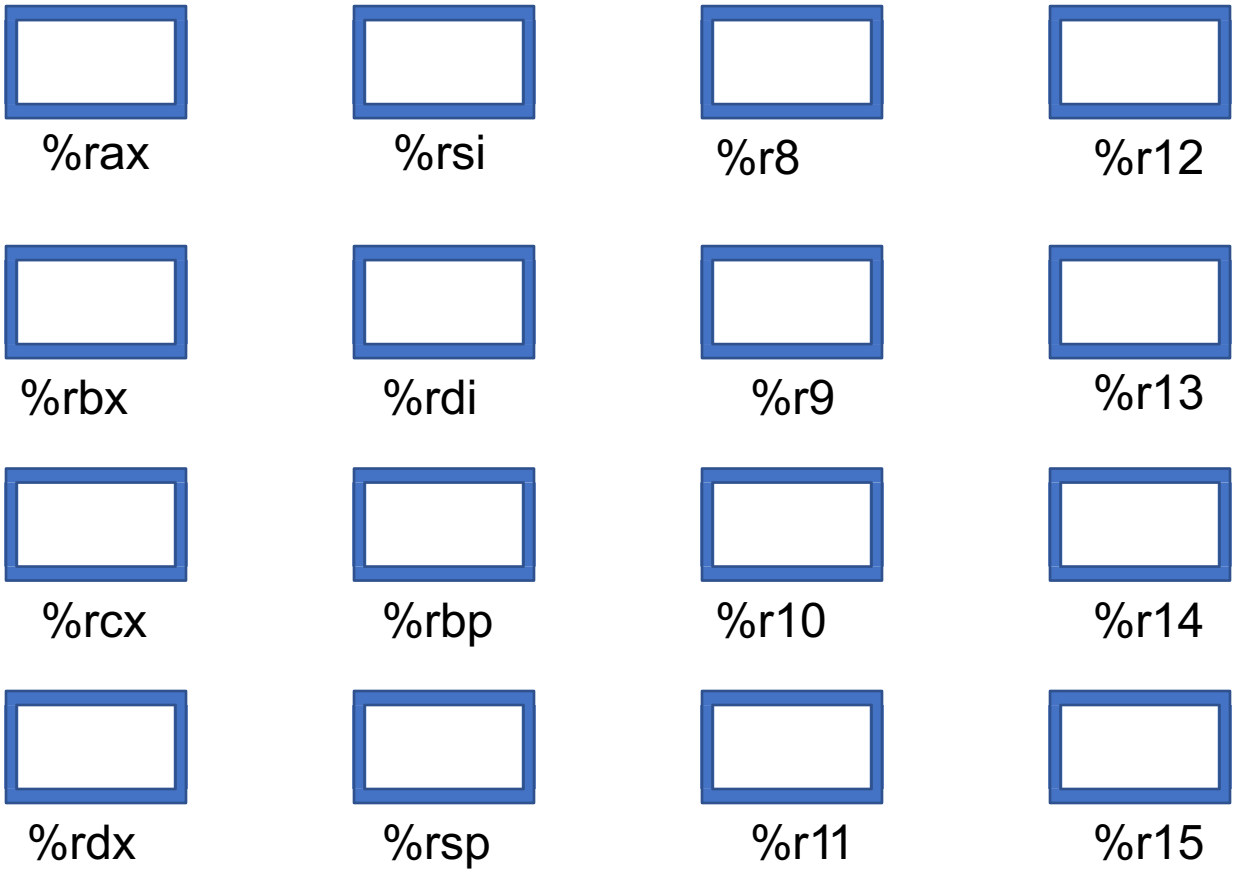
Don't forget the implicit 1:
1 + ½ + ¼

-1.75 * 2^1

= -3.5

Assembly Review

Registers



Caller/Callee

```
int main () {  //caller
    helper1 (4)
}

void helper1 (int a) {  //callee
 //function code
}
```

frame 1

frame 0

Can view frames using backtrace in gdb

Caller/callee extension

```
int main () {
    helper1 (4)
}

void helper1 (int a) {
    helper2(7)
}

void helper2 (int b) {
    //function code
}
```

Frame 2

Frame 1

Frame 0

Caller/callee cont.

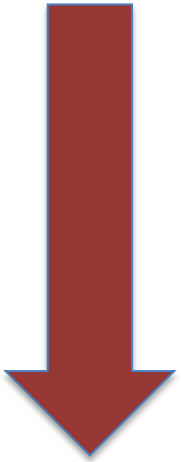
```
int main () {
    helper1 (4)
}

void helper1 (int a) {
    helper2(7)
}

void helper2 (int b) {
    //function code
}
```

Caller

Callee



Caller

Callee

Frame 2

Frame 1

Frame 0

Caller/Callee registers

- From the reference sheet:
  - callee: %r10, %r11
  - caller: %rbx, %rbp, %r12-%r15

Caller/callee at assembly level

```
int main () {
    helper1 (4)
}

void helper1 (int a) {
    helper2(7)
}

void helper2 (int b) {
    //function code
}
```

How can I preserve what's in %rdi?

I'm a callee, so I own %r10 and %r11, but I can't ise them because…

I'm about to become a caller. How can I store what's in %rdi?

I'll use caller registers! but what about main's caller register content?

Frame 2

Frame 1

Frame 0

Caller/callee at assembly level

```
int main () {
    helper1 (4)
}
void helper1 (int a) {
    helper2(7)
}
void helper2 (int b) {
    //function code
}
```

```
push %rbx  //saves main's data to
                    the stack
…
mov %rdi, %rbx  //uses caller reg
mov  $0x7, %rdi  //for calling helper2
callq  <some address> helper2
…
pop %rbx   //restores main's data
ret
```

Caller/callee at assembly level

```
int main () {
    helper1 (4)
}
void helper1 (int a) {
    helper2(7)
}
void helper2 (int b) {
    //function code
}
```

```
push %rbx  //saves main's data to
                    the stack
…
mov %rdi, %rbx  //uses caller reg
mov  $0x7, %rdi  //for calling helper2
callq  <some address> helper2
…
pop %rbx   //restores main's data
ret
```

Can freely use %r10 and %r11 without overwriting helper1's data

Generics Review

Pointer Arithmetic Review

```
print_diff(void *a, void *b ) {
    printf (“%zu\n”, (size_t) b – (size_t) a);
}
void *ptr = malloc(16);
char *char_ptr = (char *) ptr;
long *long_ptr = (long *) ptr;
int *int_ptr = (int *) ptr;

print_diff (char_ptr, char_ptr + 1);
print_diff (long_ptr, long_ptr + 1);
print_diff (int_ptr, int_ptr + 1);
print_diff (ptr, ptr + 1); //can’t do pointer arith on void *
```