An Example of Handlers I

- Let's consider an example from SUN's numerical computation guide
- In most systems there is a standard math library for functions like exp, pow, log, ...
- On SUN machines, there is an additional math library: libsunmath.a

  exp2, exp10, ..., ieee_flags, ieee_handler, ieee_retrospective
- A program:

An Example of Handlers II

```
#include <stdio.h>
#include <sys/ieeefp.h>
#include <sunmath.h>
#include <siginfo.h>
#include <ucontext.h>

void handler(int sig, siginfo_t *sip,
             ucontext_t *uap)
{
    unsigned    code, addr;
```

An Example of Handlers III

```
    code = sip->si_code;
    addr = (unsigned) sip->si_addr;
    fprintf(stderr, "fp exception %x at
        address %x \n", code, addr);
}
int main()
{
    double  x;

    /* trap on common floating point
       exceptions */
```

An Example of Handlers IV

```
if (ieee_handler("set", "common", handler)
    != 0)
    printf("Did not set exception
            handler \n");

/* cause an underflow exception (not
   reported) */
x = min_normal();
printf("min_normal = %g \n", x);
x = x / 13.0;
printf("min_normal / 13.0 = %g \n", x);
```

An Example of Handlers V

```
/* cause an overflow exception
   (reported) */
x = max_normal();
printf("max_normal = %g \n", x);
x = x * x;
printf("max_normal * max_normal = %g \n",
        x);

ieee_retrospective(stderr);
return 0;
```

An Example of Handlers VI

```
}
```

- Result:

```
min_normal = 2.22507e-308
min_normal / 13.0 = 1.7116e-309
max_normal = 1.79769e+308
fp exception 4 at address 10d0c
max_normal * max_normal = 1.79769e+308
 Note: IEEE floating-point exception flags raise
    Inexact;  Underflow;
 IEEE floating-point exception traps enabled:
```

An Example of Handlers VII

```
   overflow; division by zero; invalid operati
 See the Numerical Computation Guide, ieee_flag
ieee_handler(3M)
```

- invalid, division by zero, and overflow sometimes called common exceptions here

  ieee_handler("set", "common", handler) means handlers used for common exceptions
- min_normal / 13.0: using denormalized numbers
- handler: the subroutine to handle exceptions. Here we simply print something

The Use of Flags: An Example I

- Calculate x^n, n : integer

```
double pow(double x, int n)
{
        double tmp = x, ret = 1.0;

        for(int t=n; t>0; t/=2)
        {
                if(t%2==1) ret*=tmp;
                tmp = tmp * tmp;
        }
        return ret;
```

The Use of Flags: An Example II

```
}
```

$$x^{16} = (x^2)^8 = (x^4)^4 = \cdots$$

$x^{15} = x(x^2)^7$, treat x^2 as the new x

$$x^{15} = x(x^2)^7 = x(x^2)(x^4)^3 = x(x^2)(x^4)(x^8)^1$$

- This subrouine handles the situation if $n \geq 0$

The Use of Flags: An Example III

- If $n < 0$, we need to use

$$x^n = (1/x)^{-n} = 1/(x^{-n})$$

Example:

$$2^{-5} = (1/2)^5 = 1/(2^5)$$

- $\text{pow}(1/x, -n)$ is less accurate; $1/\text{pow}(x, -n)$ is better
- Reason: there is already error on $1/x$

The Use of Flags: An Example IV

- However, there is a small problem on using $1/\text{pow}(x, -n)$
- If $\text{pow}(x, -n)$ causes underflow (i.e. when $x < 1, n < 0$), either underflow trap handler is called or underflow status flag is set $\Rightarrow$ incorrect

  x^{-n} underflow, x^n overflow or within the range

  $(e_{\min} = -126, 2^{-e_{\min}} = 2^{126} < 2^{127} = 2^{e_{\max}})$
- Solution:

  In the beginning, turn off overflow & underflow trap enable bits, save overflow & underflow status bits

The Use of Flags: An Example V

Compute $1/\text{pow}(x, -n)$

If neither overflow nor underflow status is set $\Rightarrow$ restore them

If one is set, restore & calculate $\text{pow}(1/x, -n)$, which causes correct exception to occur

- Practically the calculation of pow() is more complicated
- In glibc-2.17/sysdeps/ieee754/dbl-64, e_pow.c has 420 lines

The Use of Flags: An Example VI

- Another example: calculate arccos using arctan

$$\arccos x = 2 \arctan \sqrt{\frac{1-x}{1+x}}$$

$$\cos\theta = x = 2\cos^2\frac{\theta}{2} - 1 = 1 - 2\sin^2\frac{\theta}{2}$$

$$\cos\frac{\theta}{2} = \sqrt{\frac{x+1}{2}}, \sin\frac{\theta}{2} = \sqrt{\frac{1-x}{2}}, \tan\frac{\theta}{2} = \sqrt{\frac{1-x}{1+x}}$$

Hence

$$\arccos x = 2 \arctan \sqrt{\frac{1-x}{1+x}}$$

The Use of Flags: An Example VII

- Consider $x = -1$

  $\arctan(\infty) = \pi/2 \Rightarrow \arccos(-1) = \pi$
- A small problem:

  $\frac{1-x}{1+x}$ causes the divide-by-zero flag set though $\arccos(-1)$ is not exceptional
- Solution: save divide-by-zero flag, restore it after arccos computation