Invariants

Eun Bae Kong
Department of Computer Science and Engineering
Chungnam National University
Daejeon, 305-764
Republic of Korea
keb@cnu.ac.kr

Integer Division

- Given X and Y, compute quotient and remainder of Y/X
- Precondition: $\{X > 0 \quad \wedge \quad Y \geq 0\}$
- Postcondition: $\{Y = qX + r \quad \wedge \quad 0 \leq r < X\}$
- Invariant: $\{Y = qX + r \quad \wedge \quad 0 \leq r\}$

Code

```
q = 0; r = Y;
while (r ≥ X) {
    q=q+1;
    r = r-X
}
```

Another Example

- Given $N > 0$, calculate $\lfloor \log_2 N \rfloor$
- Postcondition: Find e such that

$$\{2^e \leq N < 2^{e+1}\}$$

- Invariant:

$$\{k = 2^{e+1} \quad \wedge \quad 2^e \leq N\}$$

Code

```
e = -1; k = 1;
while (k ≤ N) {
    e = e+1;
    k = k*2
}
```

Another Example

- Given $N \geq 0$, calculate 2^N
- Slow way

```
prod = 1;
```

for (i=0; i $<$ N; i++) prod = 2 $\times$ prod;

Fast Way

- Idea: $2^{a+b+c} = 2^a * 2^b * 2^c$ (Actually, slow way uses $a = b = c = 1$)
- $2^{1011} = 2^{1000} * 2^{0010} * 2^{0001}$ (binary exponents)

bits	prod
1011	2^0
shift 1	$2^0 \times 2^1 = 2^1$
101	
shift 1	$2^1 \times 2^2 = 2^3$
10	
shift 0	no change
1	
shift 1	$2^3 \times 2^8 = 2^{11}$

- Keep bits in a, shift them into b.

a	b
1011	0
101	1
10	11
1	011
0	1011

- They add up to N if a is shifted left enough.

a	m	b
1011	1	0
101	2	1
10	4	11
1	8	011
0	16	1011

- $$am + b = N$$

- To maintain the above assertion while shifting a bit, we execute

```
if (odd(a)) b = b+m
m = m*2
a = a / 2
```

- We will keep the product in **prod**. (prod$=2^b$)
- When we shift a 1-bit, we must multiply prod by 2^m.
- We store 2^m in **power** to avoid exponentiation.
- Invariant:

$$\{am + b = N \quad \wedge \quad power = 2^m \quad \wedge \quad prod = 2^b \quad \wedge \quad a \geq 0\}$$

Code

```
a = N;
m = 1; power = 2;
b = 0; prod = 1;
while (a > 0) {          /* bad practice */
    if (odd(a)) {
        b = b+m; prod = prod*power;
    }
    a = a / 2;
    m = m*2;
    power = power*power;
}
```

Code

```
a = N;
m = 1; power = 2;
b = 0; prod = 1;
while (a ≠ 0) {          /* good practice */
    if (odd(a)) {
        b = b+m; prod = prod*power;
    }
    a = a / 2;
    m = m*2;
    power = power*power;
}
```