Intergalactic Electrostatic Nerf Balls

CASCON 2008 Hand's On

InterNerf Physics

- Forces
  - electrical attraction/repulsion
    - +- $1/r^2$
  - compression
    - -10/r if r < 1
  - gravity
    - 1/r

2 Loops

- iterate over all pairs (i,j) of Nerf balls, $O(n^2)$
  - calculate differences
  - calculate forces
  - calculate unit time step
  - keep track of maximum step
- iterate over NerfBalls to adjust velocity and position, O(n)

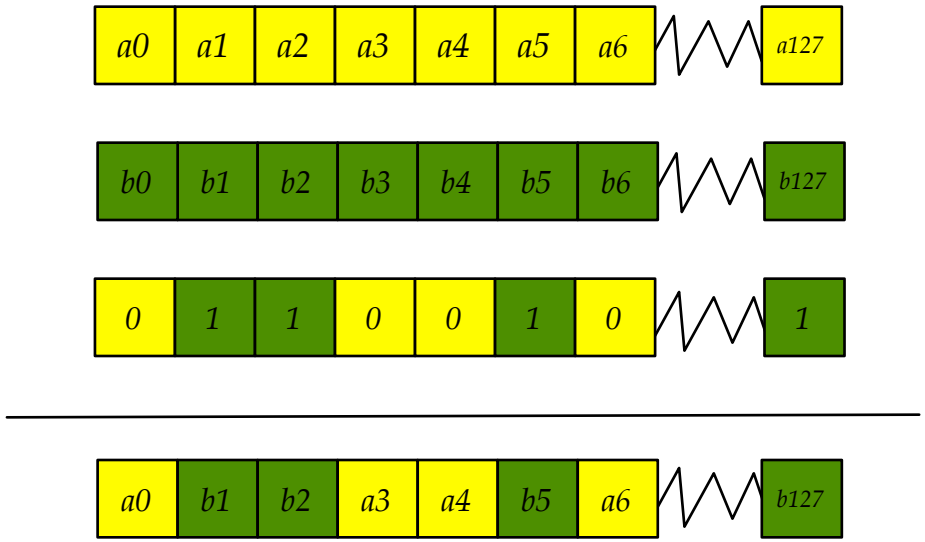
take both ways in fork

- branches are expensive
  - 22 cycles misprediction penalty
  - better to do both computations

```
diffIsBigger = si_fcgt(rSquared, maxDiff);

maxDiff = selb(maxDiff, rSquared, diffIsBigger);
```

selb



- SELect Bits
- use for
  - if without branch
  - copying sign
  - etc.

Copying Sign

- Use XOR to calculate sign parity

```
differentCharges = si_xor(c1,c2);

coulomb = selb(oneOverR2, differentCharges, signBit);
```

```
switch = si_fcgt(rSquared, (vector float){1,1,1,1};

both = coulomb + oneOverR;

force = selb(both, coulomb, switch);
```

Logic works too

```
switch = si_fcgt(rSquared, (vector float){1,1,1,1};

masked = si_and(oneOverR,switch);

force = coulomb + masked;
```

Loop Body Without Branches

```
x1 = x[i];
y1 = y[i];
z1 = z[i];
c1 = c[i];

x2 = x[j];
y2 = y[j];
z2 = z[j];
c2 = c[j];

xdiff = x1 - x2;
ydiff = y1 - y2;
zdiff = z1 - z2;

differentCharges = si_xor(c1,c2);

rSquared = fma( zdiff,zdiff, fma( ydiff,ydiff, fm(xdiff,xdiff)));

switch = si_fcgt(rSquared, (vector float){1,1,1,1};

oneOverR = rsqrtf4(rSquared);

oneOverR2 = oneOverR * oneOverR;

coulomb = selb(oneOverR2, differentCharges, (vector unsigned int){1<<31, 1<<31

both = coulomb - oneOverR;

force = selb(both, coulomb, switch);

dx[i] = force * xdiff;
dy[i] = force * ydiff;
dz[i] = force * zdiff;

dx[i] = - force * xdiff;
dy[i] = - force * ydiff;
dz[i] = - force * zdiff;

diffIsBigger = si_fcgt(rSquared, maxDiff);

maxDiff = selb(maxDiff, rSquared, diffIsBigger);
```

Still To Do

- for large problems
  - partition onto multiple SPUs
  - synchronize
- add momentum
- add wormholes
- make nifty graphics

Coconut:

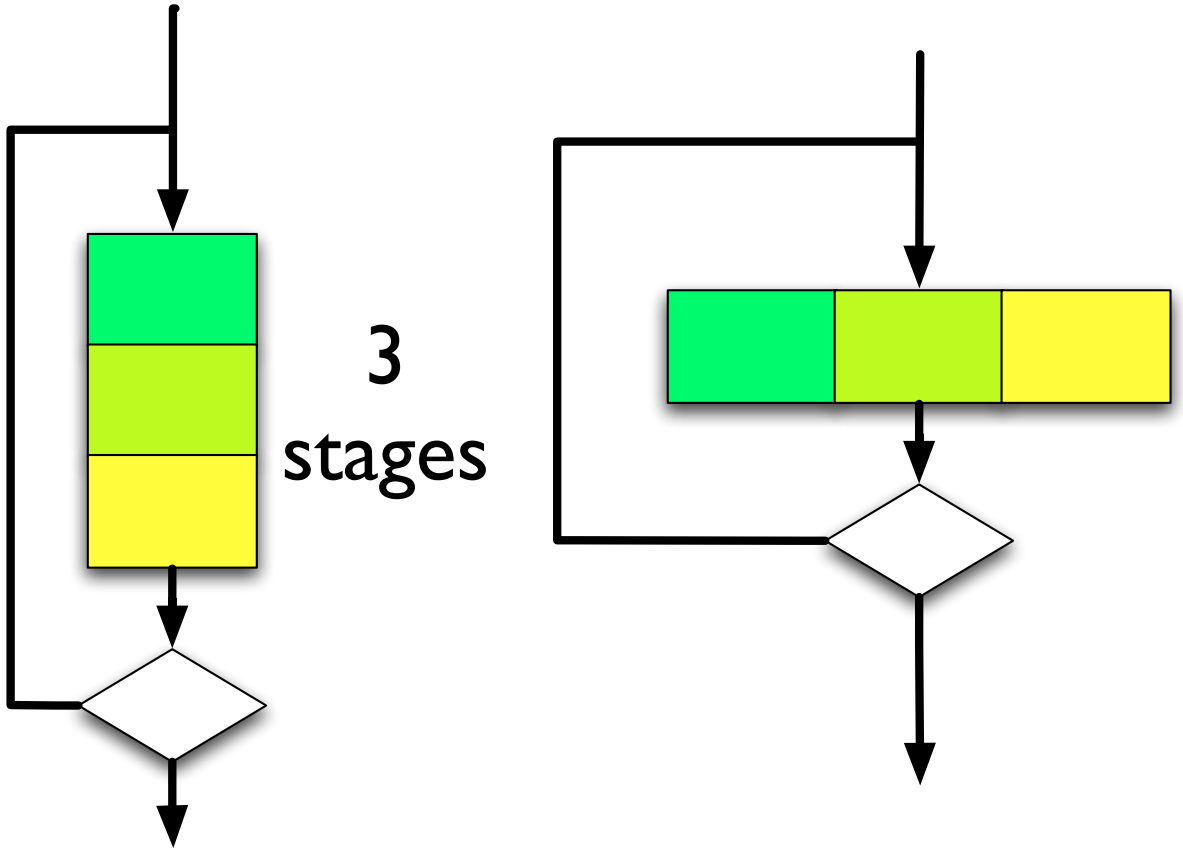
***Co*de *Con*structing *U*ser *T*ool

Christopher Kumar Anand
Wolfram Kahl

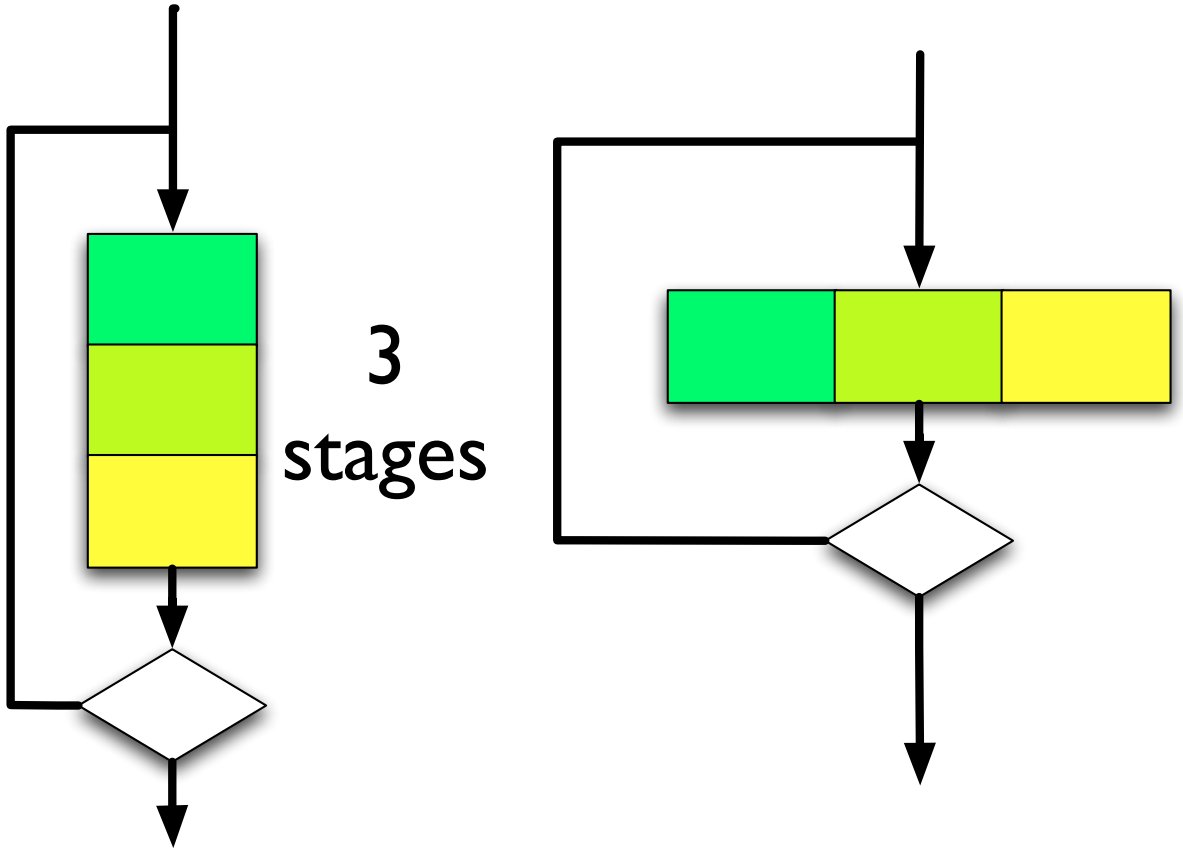
Software Quality Research Laboratory
McMaster University

http://ocalgorithms.com

Software Pipelining

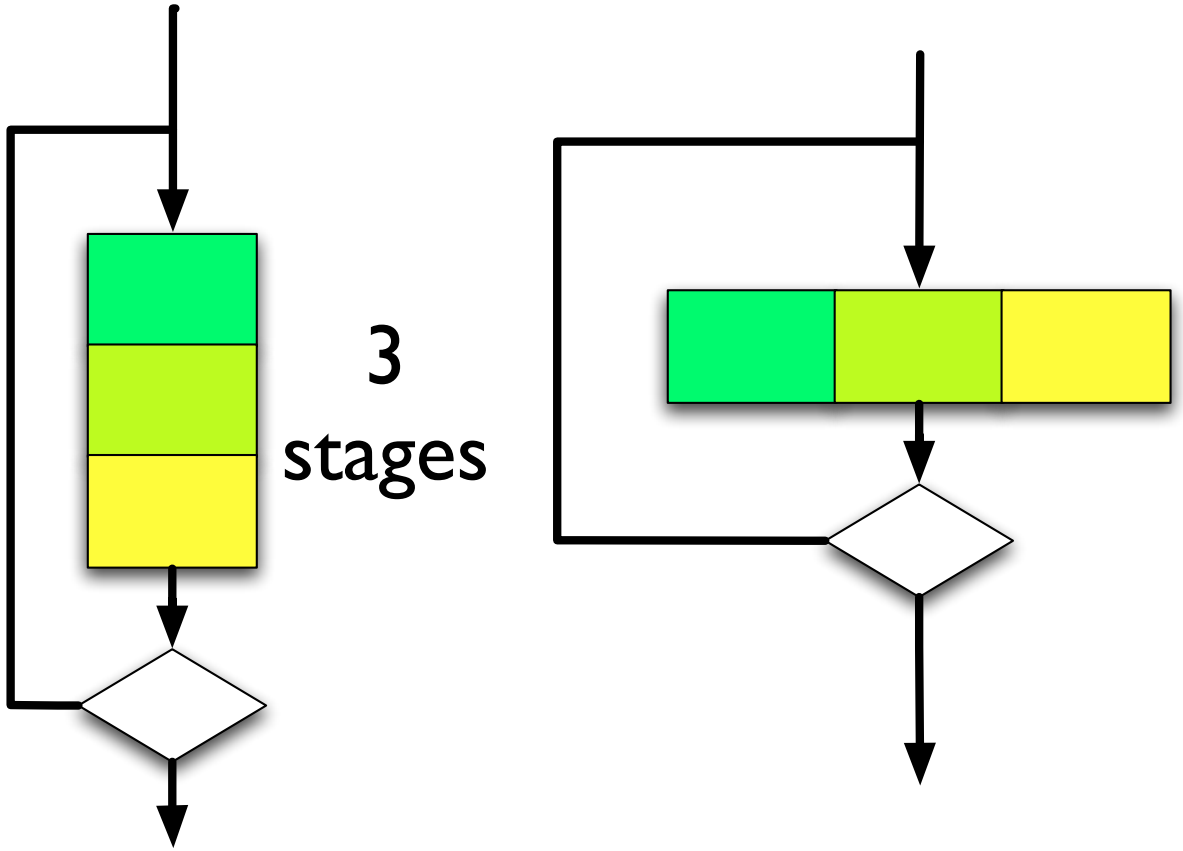


Software Pipelining



- hide latency

Software Pipelining



- hide latency
- same length loop body

Haskell Cheat Sheet

- indentation replaces parentheses
- function arguments don't take parentheses
- declarative, order-independent
- types are inferred (in correct code)

\begin{code}

code

\end{code}