NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Programming Techniques for Supercomputers Tutorial

Erlangen National High Performance Computing Center

Department of Computer Science

FAU Erlangen-Nürnberg

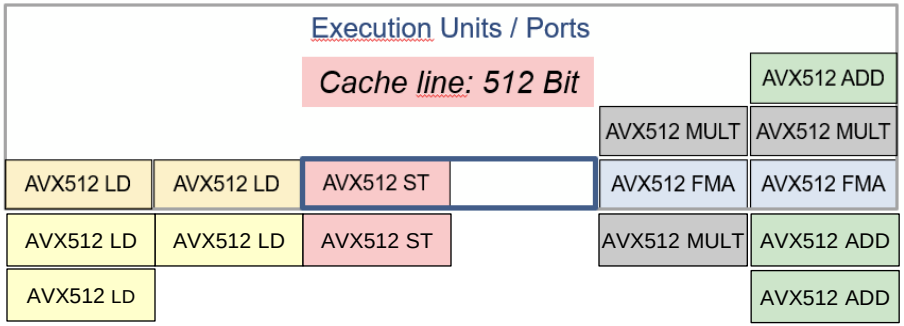
Sommersemester 2025

Assignment 3 – Task 1

Max in-core performance a) AVX512

$7[instr]/2[cy]$

```
for(int i=0; i<N; ++i)
  for(int j=1; j<M-1; ++j)
    a[i][j] = 0.2*(b[i][j] +
        b[i][j-1] + b[i][j+1]);
```



- Double precision arrays, 1.9 [GHz]

$$\boldsymbol{P_{max}} = 12\left[\frac{flops}{cy}\right] * 1.9\left[\frac{Gcy}{s}\right] = \mathbf{22.8}\left[\frac{\boldsymbol{GF}}{\boldsymbol{s}}\right]$$

$$\boldsymbol{P_{peak}} = n_{super}^{FP} * n_{FMA} * n_{SIMD} * f =$$

$$\underbrace{2\left[\frac{instr}{cy}\right] * 2 * 8\left[\frac{flops}{instr}\right]}_{32\left[\frac{flops}{cy}\right]} * 1.9\left[\frac{Gcy}{s}\right] = \mathbf{60.8}\left[\frac{\boldsymbol{GF}}{\boldsymbol{s}}\right]$$

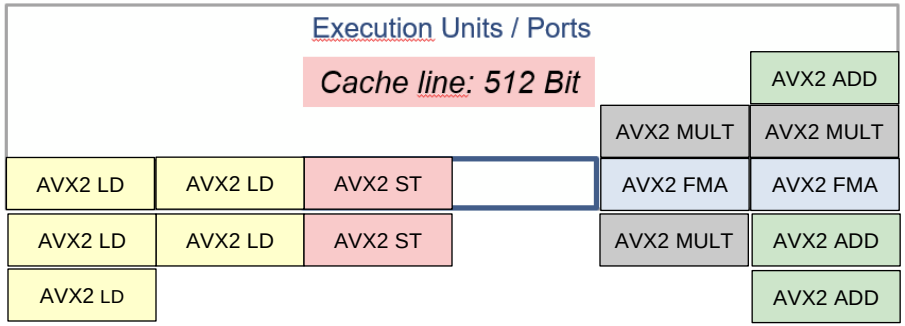
$$\Rightarrow \boldsymbol{P_{max}} = \frac{\mathbf{3}}{\mathbf{8}}\boldsymbol{P_{peak}}$$

Assignment 3 – Task 1

Max in-core performance a) AVX2

$$7[instr]/2[cy]$$

```
for(int i=0; i<N; ++i)
  for(int j=1; j<M-1; ++j)
    a[i][j] = 0.2*(b[i][j] +
        b[i][j-1] + b[i][j+1]);
```



- Double precision arrays, 1.9 [GHz]

$$\boldsymbol{P_{max}} = 6\left[\frac{flops}{cy}\right] * 1.9\left[\frac{Gcy}{s}\right] = \mathbf{11.4}\left[\frac{\boldsymbol{GF}}{\boldsymbol{s}}\right]$$

$$\Rightarrow \boldsymbol{P_{max}} = \frac{\mathbf{3}}{\mathbf{16}}\boldsymbol{P_{peak}}$$

Assignment 3 – Task 1

Max in-core performance a) Scalar

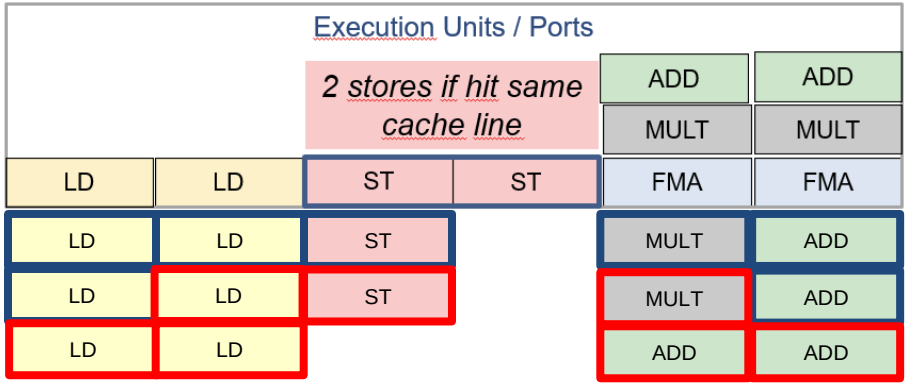
```
for(int i=0; i<N; ++i)
  for(int j=1; j<M-1; ++j)
    a[i][j] = 0.2*(b[i][j] +
        b[i][j-1] + b[i][j+1]);
```

- Double precision arrays, 1.9 [GHz]

$$\boldsymbol{P_{max}} = 2\left[\frac{flops}{cy}\right] * 1.9\left[\frac{Gcy}{s}\right] = \mathbf{3.8}\left[\frac{\boldsymbol{GF}}{\boldsymbol{s}}\right]$$

$$\Rightarrow \boldsymbol{P_{max}} = \frac{\mathbf{1}}{\mathbf{16}}\boldsymbol{P_{peak}}$$

$14[instr]/3[cy]$



Assignment 3 – Task 1

Max in-core performance b) Code Balance

```
for(int i=0; i<N; ++i)
  for(int j=1; j<M-1; ++j)
    a[i][j] = 0.2*(b[i][j] +
        b[i][j-1] + b[i][j+1]);
```

RHS: **1 RD**
LHS: **1 WR** (+ **1RD** for W/A)

- Double precision arrays
- Nontemporal stores or cache line claim cannot be used
- Assume `N` and `M` are “large” (arrays will not fit into cache)
- In each iteration, `b[i][j+1]` is read from memory
  - This element will be used from cache in the next two iterations
  - **No main memory transfers are necessary** for `b[i][j]` and `b[i][j-1]`

$$\Rightarrow \boldsymbol{B_c} = \frac{24[B/iter]}{3[flops/iter]} = \boldsymbol{8[B/F]}$$

Assignment 3 – Task 2

Latency and bandwidth

$b_s = 250\ [GB/s]$
$T_l = 130\ [ns]$

Who is the major contributor to runtime?

a) $N = 4096\ [B]$

Hockney Model: $\boldsymbol{T_{trans}} = T_l + \frac{N}{b} = 130 \times 10^{-9}[\mathrm{s}] + \frac{4096\ [B]}{250 \times 10^9 [B/s]} = \mathbf{146\ [ns]}$

$\boldsymbol{B_{eff}} = \frac{N}{T_{trans}} = 4096[B]/146[ns] = \mathbf{28}\ [\boldsymbol{GB/s}]$

b) $B_{\mathrm{eff}} = b_s/2$ with $B_{\mathrm{eff}} = \frac{N}{B}$ ➔ $\boldsymbol{N_{1/2} = T_l b_s}$

c) $\mathrm{P} = 1 + \frac{T_l}{N/b} = 1 + \frac{130\ [ns]}{64[B]/250[GB/s]} = 508.8125$ ➔ **509** prefetches needed

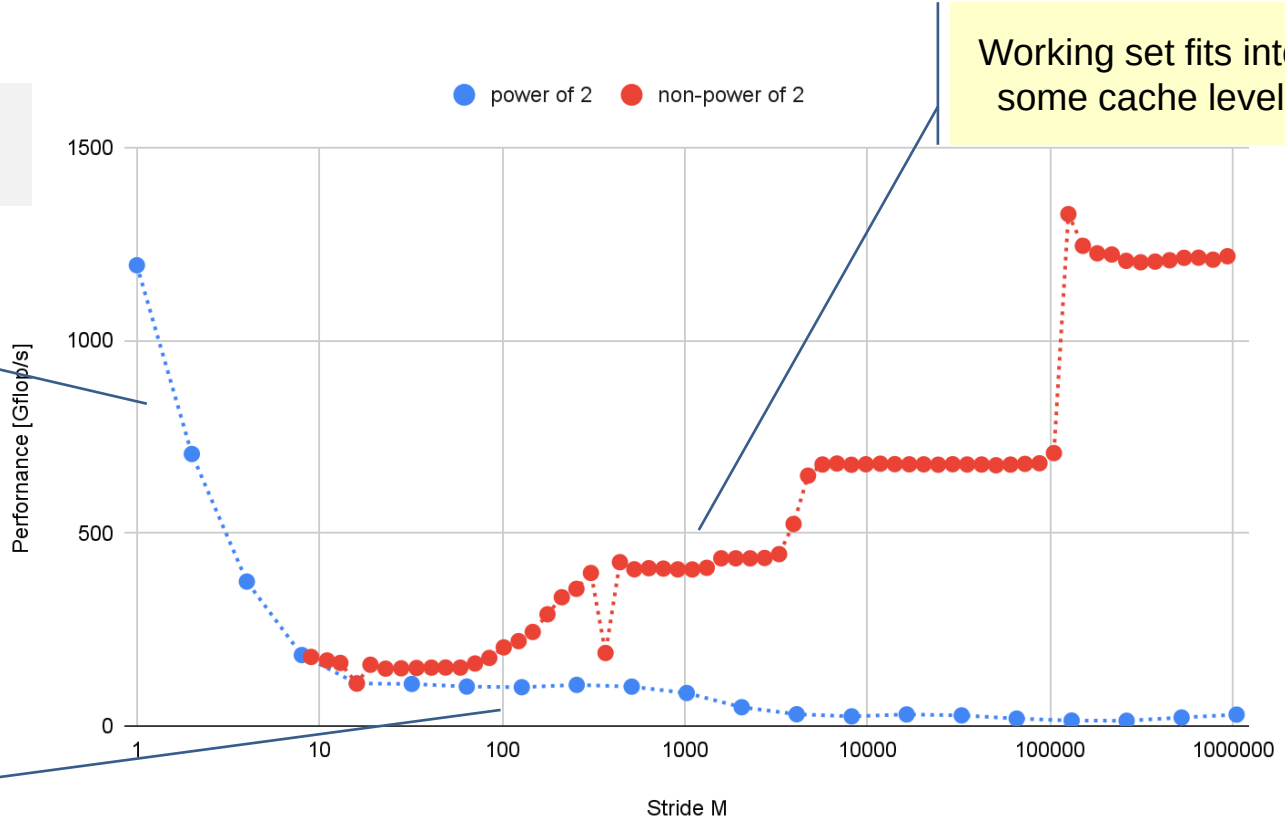
≈ 32.6 kB in-flight

Assignment 3 – Task 3

Strided access

```
for(i=0; i<N; i+=M)
    a[i] = s * a[i];
```

Only every Mth element touched



Working set fits into some cache level

Cache thrashing!