# Computer Arkitektur og Operativ Systemer

Denne forelæsning optages og gøres efterfølgende tilgængelig på Moodle
MEDDEL VENLIGST UNDERVISEREN, HVIS DU IKKE ØNSKER, AT OPTAGELSE FINDER STED

This lecture will be recorded and afterwards be made available on Moodle
PLEASE INFORM THE LECTURER IF YOU DO NOT WANT RECORDING TO TAKE PLACE

# Computer Arkitektur og Operativ Systemer

# Hukommelseshierarkiet

Lektion 8

Brian Nielsen

*Credits to*

*Randy Bryant & Dave O'Hallaron (CMU)*

# Mål

- Hvad er primær og sekundær lager?
  - Hvordan virker DRAM og Harddiske/SSD?
- Hvorfor er computerens hukommelse er modelleret som en pyramide?
- Hvordan virker cache hukommelse?
  - i dag især CPU (L1, L2, L3) caches?

```
int x[1000];
a1=x[0];
Adgangstid(x[1]) << Adgangstid(x[16])
```

- Hvorfor er program A ca. 10 gange hurtigere end program B??
  - Hvordan kan cache-venlig kode optimering øge afviklingshastighed med *10?

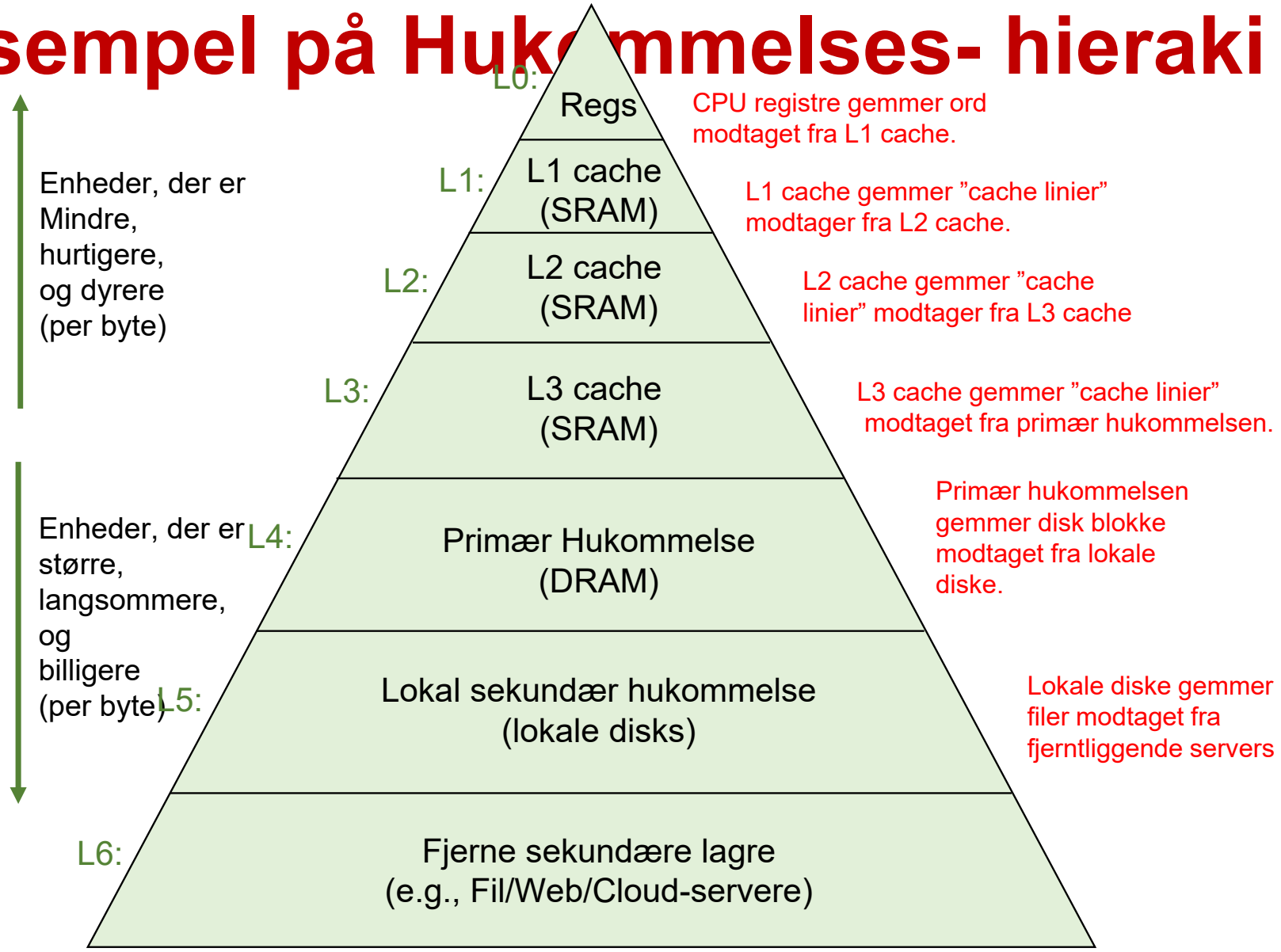
A

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

B

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

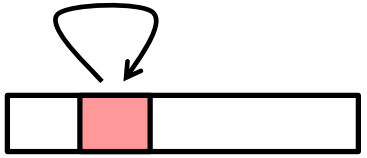
# Eksempel på Hukommelses- hieraki



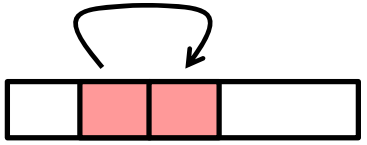
# Lokalitet – Lokalitet – Lokalitet!

- Lokalitetsprincippet: Programmer bruger (eller genbruger) ofte data og instruktioner på adresser tæt ved dem den nyligt har brugt

- Temporal lokalitet
  - Nyligt tilgået data skal sandsynligvis bruges igen i den nærmeste fremtid

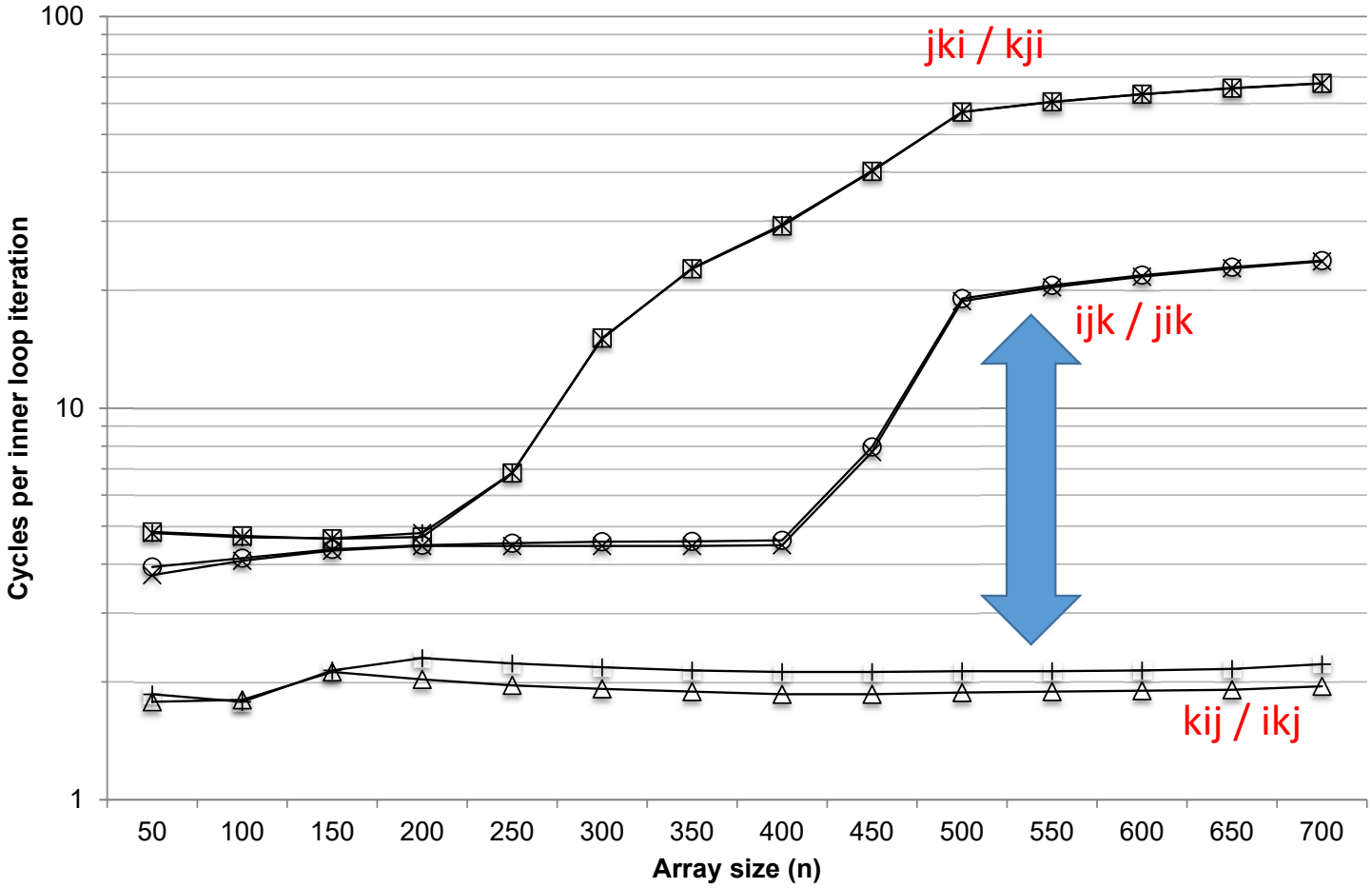


- Spatial (rummelig) lokalitet:
  - Data på nærliggende adresser refereres tit hurtigt efter hinanden



- Velskrevne programmer mod god lokalitet kører fornuftigt hurtigt
- Optimeringsteknik til programmer/alg, der skal køre rigtigt hurtigt
  - Cache-aware algoritmer (cache størrelser indgår som parameter – ”blocking”)
  - Cache-oblivious algoritmer (kører godt uafhængigt af konkrete cache parametre)

# Performance af Matrix Multiplikation på Core i7




Laveste antal miss vinder

# 1k byte hukommelse = 1024 bytes !?

NB!

- Når vi snakker størrelser på (især RAM) hukommelser, ser vi ofte
  **1k byte hukommelse = 1024 bytes "1 KB"**
- Bekvemt, at det er en potens af 2 pga. den måde vi bruger adresser på
  - Med ***n*** adresseben, kan vi adressere **$2^n$ lokationer**/celler (af 1 byte**)**
- MEN i SI systemet til fysiske enheder er 1 **kB**=1000

- IEC Alternative enheder til potenser af 2
  - 1 kibi byte 1 ***KiB*** = $2^{10}$ bytes = 1024 bytes
  - mebi = $1024^2 = 2^{20}$ =1.048.576
  - mibi = $1024^3 = 2^{30}$ , …
  - Bruges desværre ikke systematisk

**Multiple-byte units** V·T·E

| Decimal | | | Binary | | | |
|---|---|---|---|---|---|---|
| Value | Metric | | Value | IEC | | Legacy |
| 1000 | kB | **kilobyte** | 1024 | KiB | kibibyte | KB kilobyte |
| $1000^2$ | MB | megabyte | $1024^2$ | MiB | mebibyte | MB megabyte |
| $1000^3$ | GB | gigabyte | $1024^3$ | GiB | gibibyte | GB gigabyte |
| $1000^4$ | TB | terabyte | $1024^4$ | TiB | tebibyte | TB terabyte |
| $1000^5$ | PB | petabyte | $1024^5$ | PiB | pebibyte | – |
| $1000^6$ | EB | exabyte | $1024^6$ | EiB | exbibyte | – |
| $1000^7$ | ZB | zettabyte | $1024^7$ | ZiB | zebibyte | – |
| $1000^8$ | YB | yottabyte | $1024^8$ | YiB | yobibyte | – |

Orders of magnitude of data

https://en.wikipedia.org/wiki/Kilobyte

# Øvelserne

- Lokalitet i programmer:
  - Gennemløb m. skridtlængde 1.
  - Identificer alg. variant som udviser bedst spatial lokalitet.
  - Lokalitet og  CPU caches med matrix transponering
- Virkemåde af CPU caches (associative)
- (Hvornår bliver en SSD slidt op??)
- Challenge 9: Er din DRAM følsom for Row-hammer angreb?
- Challenge 10:  Udlæs cache-systemet for din maskine!

Brug Hjælpelærer assistancen! Evt også til spm. mm lektionen