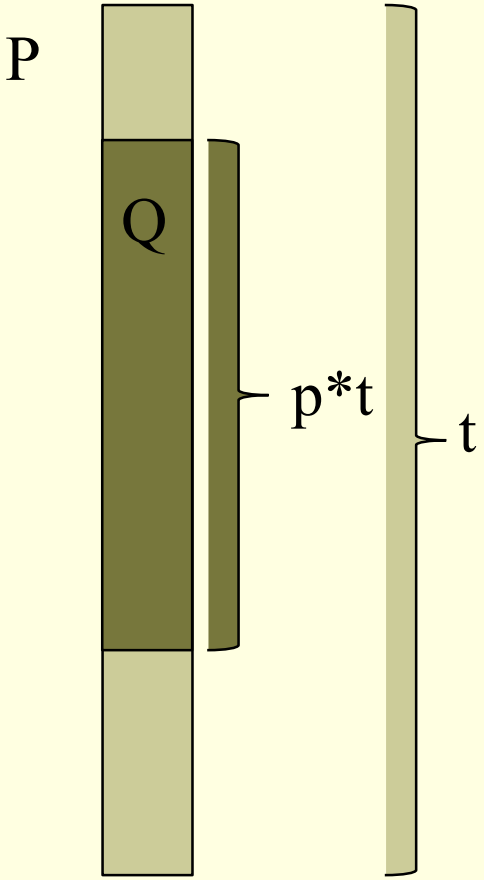
Sadržaj:

- Alati metode za poboljšavanje performansi
  - Profileri
  - Biblioteke
  - Programski jezici
- VTune Amplifier
- Integrated Performance Primitives IPP
- Domaće zadaće 3 i 4

Osnove profiling-a

Vrijeme izvođenja programa P je t
Dio programa Q u omjeru p se može ubrzati N puta
Koliko je ubrzanje cijelog programa U?



Amdahlov zakon:

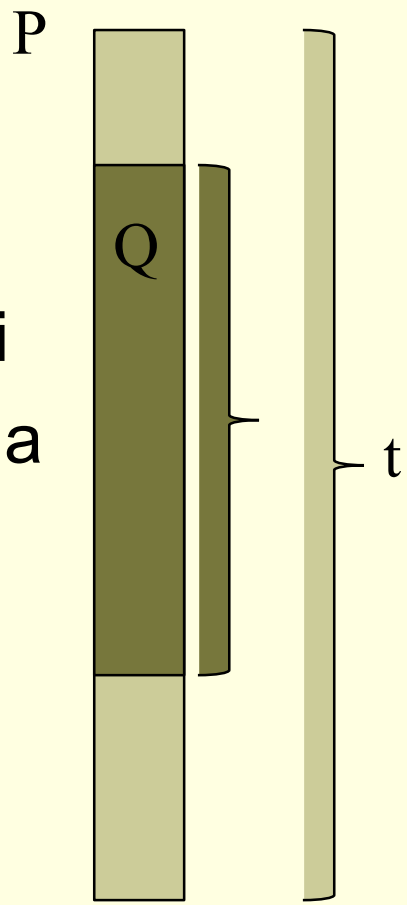
$$U = \frac{t}{t'} = \frac{t}{t\left((1-p) + \frac{p}{N}\right)} = \frac{1}{\left(1 - p + \frac{p}{N}\right)}$$

Ako $N \longrightarrow \infty$

$$U \approx \frac{1}{1-p}$$

Osnove profiling-a

- Zašto je Amdahl-ov zakon bitan
  - Jer definira realnu situaciju u kojoj se dio programa ne može ubrzati, a postoje i dijelovi koji se mogu ubrzati
  - Definira maksimalnu granicu ubrzanja
  - Pomaže u definiranju strategije ubrzavanja programa



Osnove profiling-a

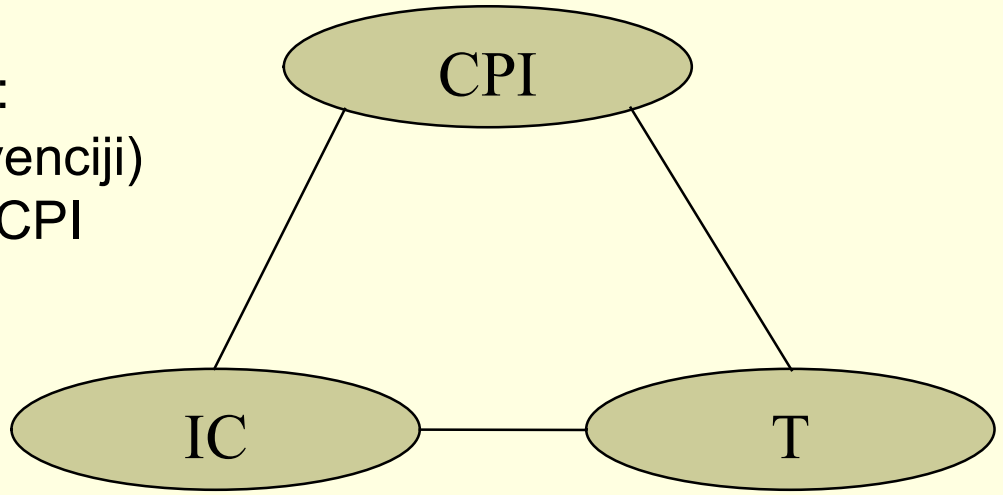
1. Cilj profiliranja je pronaći dijelove programa koji troše najviše resursa (usko grlo)
   - Vrijeme
   - Energija
   - Sklopovski: memorija, dma, disk, sabirnica
2. Definirati strategiju ubrzavanja identificiranih dijelova
3. Strategije ubrzavanja:
   - Asembler (SIMD-izacija)
   - Biblioteke (IPP, MKL, boost, BLAS, LAPACK)
   - Paralelizam (TBB, OpenMP, CUDA, OpenCl, MPI)
   - Novi pristupi (DSL, Go, Cilk+)

Osnove profiling-a

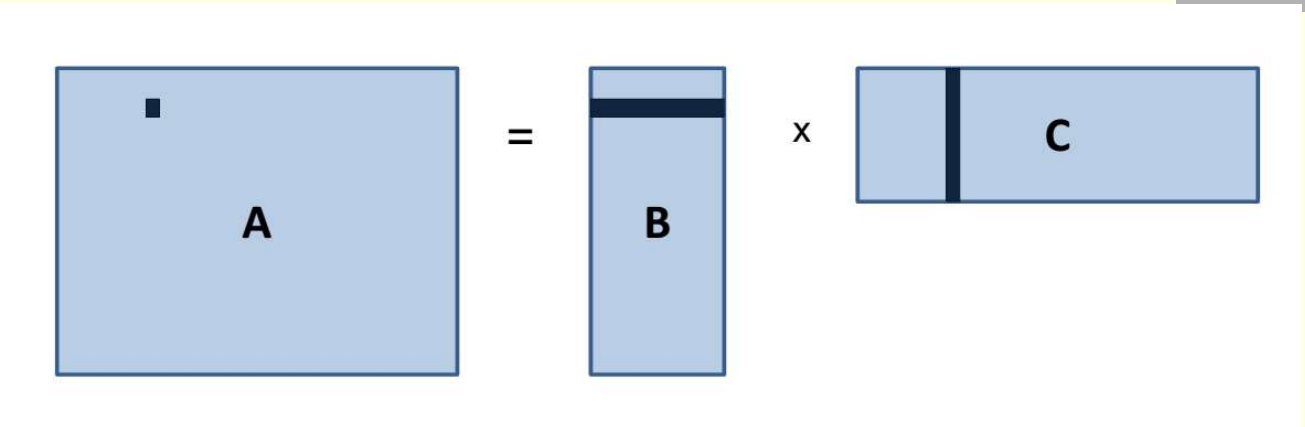
$$T = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clockcycle}} = \text{CPU time}$$

Računske performanse ovise o:

- Vremenskom taktu T (frekvenciji)
- Broju ciklusa po instrukciji CPI
- Broju instrukcija IC



Osnove: Množenje matrica

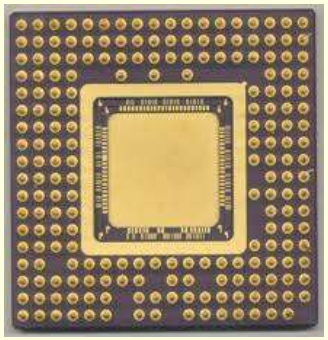


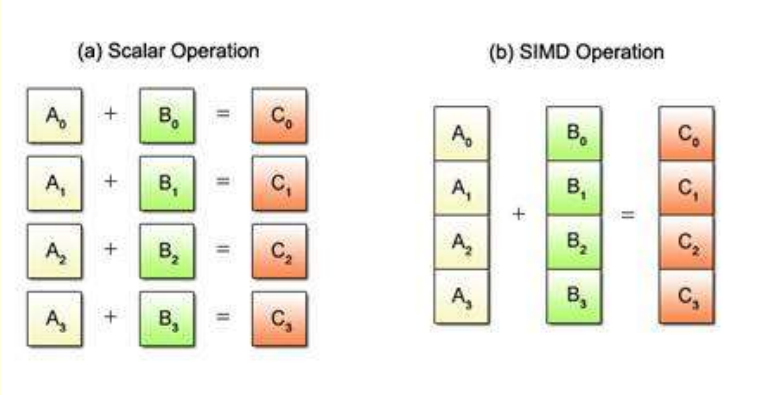

```
for(int i = 0; i < x; i++)
    for(int j = 0; j < y; j++)
        for(int k=0; k < z; k++)
            A[i][j] += B[i][k] * C[k][j];
```

- Matrica B: pristup po retcima
- Matrica C: pristup po stupcima

Jednoprocesorski paralelizam

- ILP – Instruction Level Parallelism
  - Arhitekti računala skrivaju paralelizam (2002)
  - Protočna arhitektura (preklapanje dijelova instrukcija)
  - Superskalarno izvođenje (obrada više instrukcija u isto vrijeme)
  - VLIW: Prevoditelj određuje ILP
  - Vektorska obrada: Grupe sličnih nezavisnih operacija (SIMD)
  - “Out of Order Execution”: Instrukcije se prividno izvode po redu u toku instrukcija, stvarno ovisno o slobodnim resursima



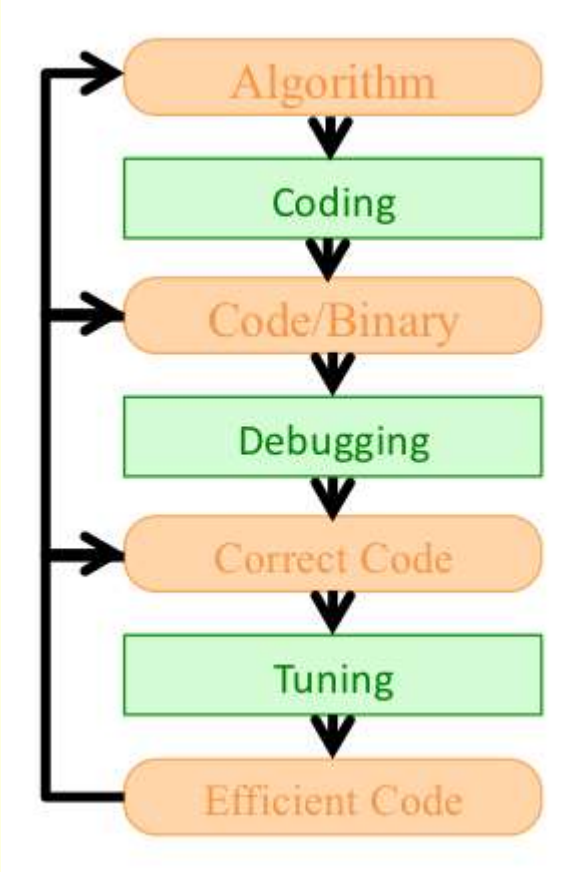


Ograničenja ILP-a

- Hazardi:
  - Strukturni – računski resursi
  - Podatkovni – međuovisnost podataka
  - Kontrolni – Upravljanje tijekom izvođenja
- Granice (Wall)
  - Power wall
  - ILP wall
  - Memory wall

Performance profiling

- Razvojni ciklus
- Tuning:
  - Mjerenje performansi
  - Analiza mjerenja
  - Modifikacija algoritma
  - Ponovno mjerenje
  - Analiza razlika



Kako mjeriti performanse

- Mjerenje vremena izvođenja
  - Načelno
  - Ne mogu se otkriti kritične točke
- Integracija programskog koda za mjerenje
  - C/C++: clock_t, clock() iz <time.h>
  - Nije pogodno za održavanje
- Korištenja alata za profiling
  - Vtune Amplifier, gprof, …
  - Detaljna analiza
  - Povezanost sa izvornim kodom

Tipovi *profiling*-a

- Statističko uzorkovanje (*statistical sampling*)
  - Periodičko prekidanje izvođenja i pamćenje lokacije
  - Analiza statističke distribucije
  - Vremensko skupljanje podataka (vrijeme)
  - Uniforman *overhead*
- Praćenje događaja (*event tracing*)
  - Skupljanje individualnih događaja (pozivi funkcija, razmjena poruka)
  - Detaljna analiza događaja
  - Velika količina podataka i *overhead*

Gnu profiler: gprof

- Unix/Linux
  - gcc –pg program.cpp –o program
  - ./program
  - gprof program > program.prof

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 92.63      2.75     2.75     4096   671.68   671.68  dct()
  2.69      2.83     0.08    12288     6.51     6.51  pipe(rlstruct*, pestruct*)
  1.68      2.88     0.05     4096    12.21    12.21  rgb2yuv()
  1.35      2.92     0.04    12288     3.26     3.26  runlen(int*, rlstruct*, int)
  0.67      2.94     0.02   823296     0.02     0.04  codeinsert(int)
  0.34      2.95     0.01   823296     0.01     0.01  status(pestruct*, int*, int)
  0.34      2.96     0.01   113555     0.09     0.09  order(char*, int*, int, int, int*)
  0.34      2.97     0.01     4096     2.44     2.44  getblock(int)
  0.00      2.97     0.00    12288     0.00     0.00  zigzag(int*)
  0.00      2.97     0.00    12288     0.00    13.03  entropy(int*, int)
```

GNU profiler: gprof

- *Flat profile*
  - Provedeno vrijeme u svakoj funkcij
  - Broj poziva funkcija
  - Jednostavan pronalazak vremenski kritičnih funkcija
- *Call graph*
  - Analiza po funkcijama (tko je pozvao, koga sam pozvao, koliko puta)
  - Procjena koliko vremena je provedeno u pozvanim funkcijama
  - Potencijalna mjesta za uklanjanje poziva

Vtune Amplifier

- Performanse CPU, GPU
- Skalabilnost, Propusnost, Dretve
- Vizualizacija rezultata
- *Hotspot*
  - Mjesto u programu s izraženom aktivnosti
    - Vrijeme
    - Pristup memoriji

Vtune Amplifier

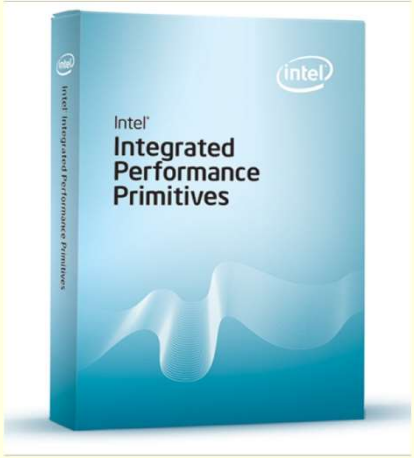
- Identifikacija *hotspot-ova*
- Identifikacija neučinkovitih dijelova programa
- Identifikacija dijelova koji su pogodni za optimizaciju
- Analiza sinkornizacijskih objekata koji utječu na iskorištenost sustava
- Analiza I/O operacija
- Aktivnost dretvi i tranzicije
- Sklopovski kritične točke u programu

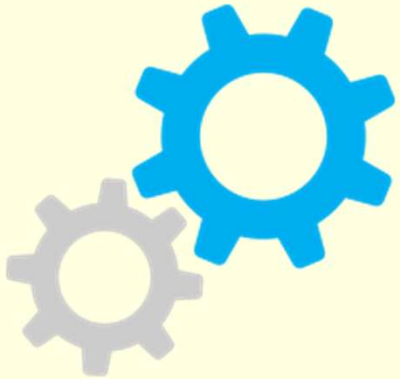
Terminologija

- **Target***:* izvršni program koj se analizira
- **Baseline:** osnovna mjera i model koji se koristi za usporedbu
- **CPU time:** vrijeme kojeg program izvodi na procesoru (za više dretvi, CPU vremena svih dretvi su zbrojena u programsko CPU vrijeme)
- **Elapsed time:** ukupno vrijeme (wall clock time) potrebno za izvođenje programa
- **Hotspot:** Dio programa sa značajnim doprinosom u ukupnom vremenu izvođenja programa
- **CPI rate:** broj taktova po instrukciji (*clocks per instruction)*

Integrated Performance Primitives IPP

- Biblioteka optimiranih funkcija za multimediju, obradu i kodiranje audio i video podataka, vektorsku manipulaciju, konverziju, kriptografiju itd.
- Zasniva se na apstrakciji procesorskih svojstava kao što su MMX, SIMD, SSEx.
- Nedostatak: Ovisnost o arhitekturi procesora i ekstenzija





Integrated Performance Primitives IPP

Code of Domain	Header file	Prefix	Description
CC	ippcc.h	ippi	color conversion
CH	ippch.h	ipps	string operations
CORE	ippcore.h	ipp	core functions
CP	ippcp.h	ipps	cryptography
CV	ippcv.h	ippi	computer vision
DC	ippdc.h	ipps	data compression
I	ippi.h	ippi	image processing
S	ipps.h	ipps	signal processing
VM	ippvm.h	ipps	vector math
E*	ippe.h	ipps	embedded functionality

* available only within the Intel® System Studio suite

Intel IPP

- Imenovanje funkcija:

ipp<data-domain><name>_<datatype>[_<descriptor>](<parameters>);

- Primjer:

```
ippiRGBToYUV_8u_C3R();
```

Tip podataka

Naziv funkcije: Pretvorba RGB u YUV

Format zapisa podataka

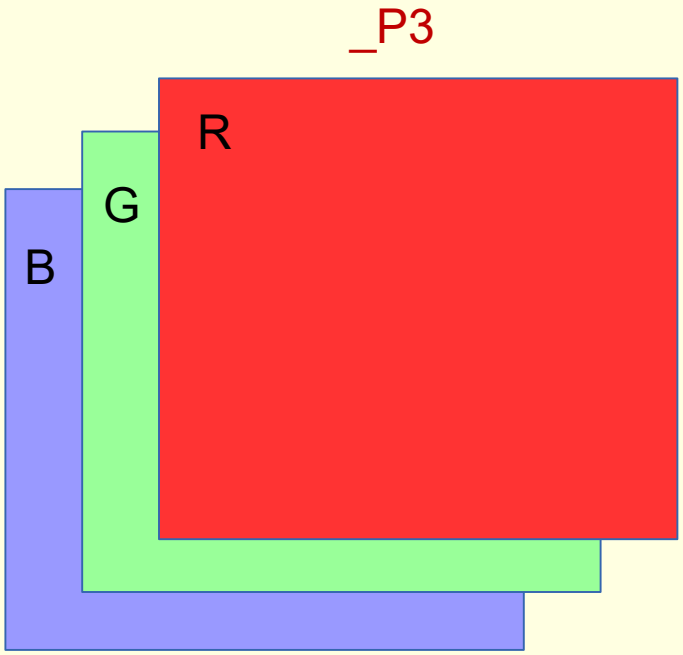
IPP funkcija, i: image processing

Intel IPP – Image processing

- Format zapisa: raspored komponenti slikovnih podataka
  - Kanalni raspred (Channel Data Layout)
    - Oznaka: “_C”
  - Planarni format (Planar Data Layout)
    - Oznaka: “_P”

_C3

RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB



Primjer: RGB → YUV

- IppStatus ippiRGBToYUV_<mod>

  (const Ipp8u* pSrc, int srcStep, Ipp8u* pDst, int dstStep, IppiSize roiSize);

  - <mod>: 8u_C3R, 8u_AC4R
- IppStatus ippiRGBToYUV_8u_P3R

  (const Ipp8u* const pSrc[3], int srcStep, Ipp8u* pDst[3], int dstStep, IppiSize roiSize);
- IppStatus ippiRGBToYUV_8u_C3P3R

  (const Ipp8u* pSrc, int srcStep, Ipp8u* pDst[3], int dstStep, IppiSize roiSize);

IPP – Korisne funkcije

IppStatus ippiDCT8x8FwdLS_<mod>()

IppStatus ippiDCT8x8Inv_<mod>()

Više u opsežnoj dokumentaciji

https://software.intel.com/content/www/us/en/develop/documentation/ipp-dev-reference/top.html