Arhitektura računala 2

moderna arhitektura računala

Siniša Šegvić

Slobodan Ribarić

Zavod za elektroniku, mikroelektroniku,

računalne i inteligentne sustave

Fakultet elektrotehnike i računarstva

Sveučilište u Zagrebu

Uvod: podsjetimo se na prvi ciklus

Prvi ciklus — klasična arhitektura računala:

- Von Neumannovo računalo
- Pojednostavljeni model procesora
- Upravljanje slijedom instrukcija
- Ožičena izvedba upravljačke jedinice
- Mikroprogramirana izvedba upravljačke jedinice

Uvod: što ćemo razmatrati sada?

Drugi ciklus — temelji moderne arhitekture računala

- Performansa, klasifikacije arhitektura,
  prostor oblikovanja procesora opće namjene
- instrukcijska arhitektura RISC procesora,
  prevođenje i pokretanje programa
- Aritmetička jedinica procesora
- Put podataka arhitekture MIPS
- Protočnost

PERFORMANSA: VRIJEME ODZIVA I PROPUSNOST

Prisjetimo se, arhitektura računala razmatra **performansu**, cijenu, utrošak energije, pouzdanost, raspoloživost

Dvojako značenje performanse računala:

1. vrijeme odziva (za sada razmatramo upravo to!):
   - koliko vremena je potrebno za obavljanje zadatka?
   - uglavnom relevantno kod radnih stanica i ugrađenih primjena
2. propusnost:
   - ukupni obavljeni posao u jedinici vremena (npr, transakcija/min)
   - uglavnom relevantno kod poslužitelja

PERFORMANSA: VRIJEME ODZIVA I PROPUSNOST

Prisjetimo se, arhitektura računala razmatra **performansu**, cijenu, utrošak energije, pouzdanost, raspoloživost

Dvojako značenje performanse računala:

1. vrijeme odziva (za sada razmatramo upravo to!):
   - koliko vremena je potrebno za obavljanje zadatka?
   - uglavnom relevantno kod radnih stanica i ugrađenih primjena
2. propusnost:
   - ukupni obavljeni posao u jedinici vremena (npr, transakcija/min)
   - uglavnom relevantno kod poslužitelja

Kako na performansu utječe zamjena procesora bržim modelom?

A nabava dodatnog procesora?

PERFORMANSA: DEFINICIJA I RELATIVNA PERFOMANSA

Performansa $P \triangleq$ (vrijeme izvođenja)$^{-1}$

Performansa: definicija i relativna perfomansa

Performansa $P \triangleq$ (vrijeme izvođenja)$^{-1}$

Što znači da računalo A ima n puta veću performansu od računala B?

$$\frac{P_A}{P_B} = n \Rightarrow \frac{t_B}{t_A} = n$$

PERFORMANSA: DEFINICIJA I RELATIVNA PERFOMANSA

Performansa $P \triangleq$ (vrijeme izvođenja)$^{-1}$

Što znači da računalo A ima n puta veću performansu od računala B?

$$\frac{P_A}{P_B} = n \Rightarrow \frac{t_B}{t_A} = n$$

Najčešće će nas interesirati upravo relativna performansa p:

$$p_A(B) \triangleq \frac{P_A}{P_B} = \frac{t_B}{t_A}$$

PERFORMANSA: DEFINICIJA I RELATIVNA PERFOMANSA

Performansa $P \triangleq$ (vrijeme izvođenja)$^{-1}$

Što znači da računalo A ima n puta veću performansu od računala B?

$$\frac{P_A}{P_B} = n \Rightarrow \frac{t_B}{t_A} = n$$

Najčešće će nas interesirati upravo relativna performansa p:

$$p_A(B) \triangleq \frac{P_A}{P_B} = \frac{t_B}{t_A}$$

Npr, izmjerimo trajanje izvođenja programa X na računalima A i B:

- $t_A(X) = 10\,\mathrm{s}$, $t_B(X) = 15\,\mathrm{s}$
- $\Rightarrow p_A(B, X) = \frac{t_B(X)}{t_A(X)} = 1{,}5$
- Računalo A je za program X $1{,}5$ puta brže!

Performansa: Amdahlov zakon

Amdahlov zakon opisuje poboljšanje performanse uslijed ubrzavanja samo jednog segmenta obrade. Neka je zadano:

- s faktor ubrzanja promatranog segmenta obrade
- x udio razmatranog dijela u cijelom postupku

Tada vrijedi:

$$p = \frac{T_{\text{staro}}}{(1-x) \cdot T_{\text{staro}} + x/s \cdot T_{\text{staro}}} = \frac{1}{(1-x) + x/s}$$

Performansa: Amdahlov zakon

Amdahlov zakon opisuje poboljšanje performanse uslijed ubrzavanja samo jednog segmenta obrade. Neka je zadano:

- s faktor ubrzanja promatranog segmenta obrade
- x udio razmatranog dijela u cijelom postupku

Tada vrijedi:

$$p = \frac{T_{\text{staro}}}{(1-x) \cdot T_{\text{staro}} + x/s \cdot T_{\text{staro}}} = \frac{1}{(1-x) + x/s}$$

Primjer: pretpostavimo da se u nekom algoritmu na množenje troši 80% vremena.

Koliko trebamo poboljšati množenje ako želimo algoritam ubrzati 5 puta?

Performansa: Amdahlov zakon

Amdahlov zakon opisuje poboljšanje performanse uslijed ubrzavanja samo jednog segmenta obrade. Neka je zadano:

- s faktor ubrzanja promatranog segmenta obrade
- x udio razmatranog dijela u cijelom postupku

Tada vrijedi:

$$p = \frac{T_{\text{staro}}}{(1-x) \cdot T_{\text{staro}} + x/s \cdot T_{\text{staro}}} = \frac{1}{(1-x) + x/s}$$

Primjer: pretpostavimo da se u nekom algoritmu na množenje troši 80% vremena.

Koliko trebamo poboljšati množenje ako želimo algoritam ubrzati 5 puta?

Odgovor: zadatak nije moguće riješiti!

Performansa: Amdahlov primjer

Razmatramo nabavu novog procesora za 8 godina stari produkcijski poslužitelj:

- poznato je da poslužitelj 60% vremena čeka na diskove i mrežu
- performansa novog procesora je $10\times$ veća od starog
- koliki bi bio učinak akvizicije?

Performansa: Amdahlov primjer

Razmatramo nabavu novog procesora za 8 godina stari produkcijski poslužitelj:

- poznato je da poslužitelj 60% vremena čeka na diskove i mrežu
- performansa novog procesora je $10\times$ veća od starog
- koliki bi bio učinak akvizicije?

$$p = \frac{1}{(1-x) + x/s} = \frac{1}{0.6 + 0.4/10} \doteq 1.56$$

Performansa: Amdahlov primjer

Razmatramo nabavu novog procesora za 8 godina stari produkcijski poslužitelj:

- poznato je da poslužitelj 60% vremena čeka na diskove i mrežu
- performansa novog procesora je $10\times$ veća od starog
- koliki bi bio učinak akvizicije?

$$p = \frac{1}{(1-x) + x/s} = \frac{1}{0.6 + 0.4/10} \doteq 1.56$$

Odgovor: 56%.

Performansa: Amdahlov primjer

Razmatramo nabavu novog procesora za 8 godina stari produkcijski poslužitelj:

- poznato je da poslužitelj 60% vremena čeka na diskove i mrežu
- performansa novog procesora je $10\times$ veća od starog
- koliki bi bio učinak akvizicije?

$$p = \frac{1}{(1-x) + x/s} = \frac{1}{0.6 + 0.4/10} \doteq 1.56$$

Odgovor: 56%.

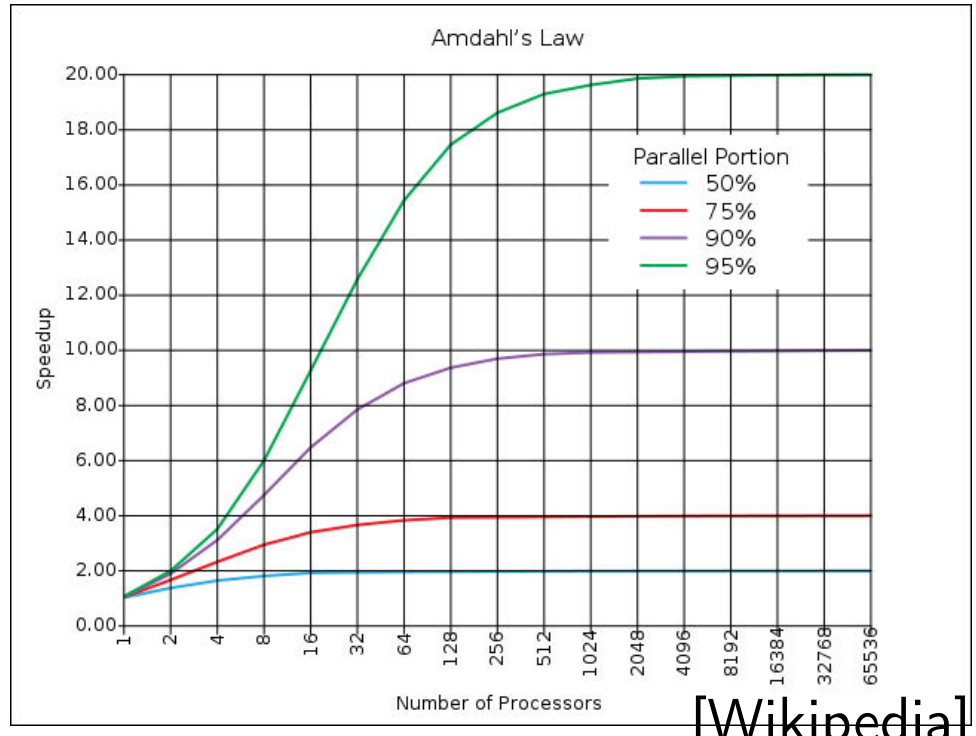
Pokazuje se da kod ovakvih proračuna ljudi često griješe.
Amdahlov zakon nam pomaže da ostvarimo bolji kontakt sa stvarnošću.

PERFORMANSA: AMDAHL ZA PARALELNE IZVEDBE

Koliko možemo ubrzati algoritam na paralelnom računalu ako slijedni udio iznosi w?

Amdahlov zakon: maksimalno $1/w$.

Desno: graf ubrzanja u ovisnosti o broju procesora s



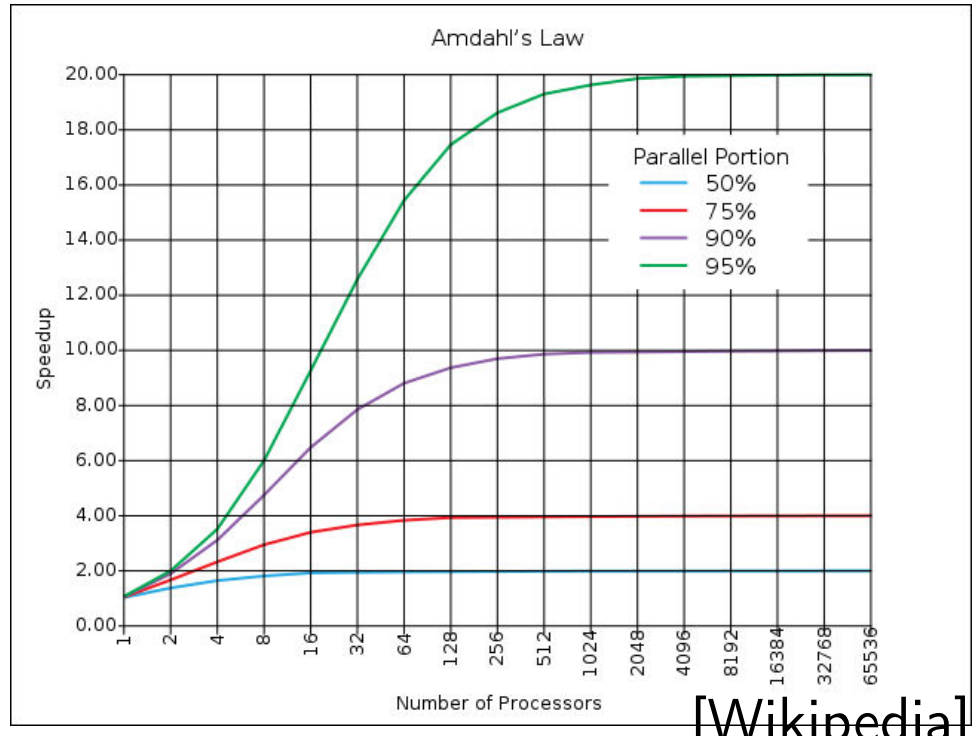
[Wikipedia]

Performansa: Amdahl za paralelne izvedbe

Koliko možemo ubrzati algoritam na paralelnom računalu ako slijedni udio iznosi w?

Amdahlov zakon: maksimalno $1/w$.

Desno: graf ubrzanja u ovisnosti o broju procesora s



[Wikipedia]

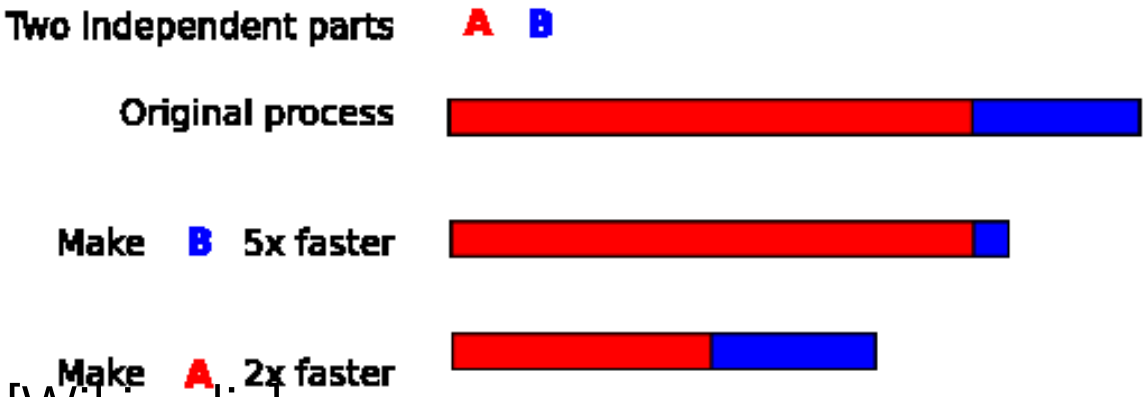
Ograničenja ovog modela:

- ukupan posao u paralelnom algoritmu je tipično veći nego u ekvivalentnom slijednom postupku
- ipak, performansa paralelnog sustava može rasti i superlinearno zbog veće količine akumulirane priručne memorije

Performansa: posljedice Amdahla

Amdahlov zakon kaže da ne možemo očekivati ukupno poboljšanje proporcionalno poboljšanju samo jednog aspekta obrade.

Glavna posljedica Amdahlovog zakona:
resurse valja alocirati za poboljšanje najčešćeg slučaja
(engl. make common case fast)



[Wikipedia]

PERFORMANSA: O MJERENJU TRAJANJA IZVOĐENJA

Više načina za mjerenje vrijeme izvođenja programa

- ukupno vrijeme izvođenja uključuje sve aspekte obrade:
  - najizravnija mjera performanse, determinira performansu sustava
  - čimbenici: CPU, RAM, U-I (mreža, disk), OS, ...

PERFORMANSA: O MJERENJU TRAJANJA IZVOĐENJA

Više načina za mjerenje vrijeme izvođenja programa

- ukupno vrijeme izvođenja uključuje sve aspekte obrade:
  - najizravnija mjera performanse, determinira performansu sustava
  - čimbenici: CPU, RAM, U-I (mreža, disk), OS, ...
- trajanje izvođenja procesa određeno performansom procesora
  - zanemaren utjecaj U-I i ostalih procesa
  - vrijeme procesa obuhvaća:
    - vrijeme korisničkog programa t_{CPU}
    - vrijeme operacijskog sustava t_{OS}
    - za detalje pogledati dokumentaciju naredbe time na UNIX-ima
  - različiti programi imaju različit omjer t_{CPU}/t_{OS}, ali najčešće je ipak $t_{\mathrm{CPU}} \gg t_{OS}$

PERFORMANSA: O MJERENJU TRAJANJA IZVOĐENJA

Više načina za mjerenje vrijeme izvođenja programa

- ukupno vrijeme izvođenja uključuje sve aspekte obrade:
  - najizravnija mjera performanse, determinira performansu sustava
  - čimbenici: CPU, RAM, U-I (mreža, disk), OS, ...
- trajanje izvođenja procesa određeno performansom procesora
  - zanemaren utjecaj U-I i ostalih procesa
  - vrijeme procesa obuhvaća:
    - vrijeme korisničkog programa t_{CPU}
    - vrijeme operacijskog sustava t_{OS}
    - za detalje pogledati dokumentaciju naredbe time na UNIX-ima
  - različiti programi imaju različit omjer t_{CPU}/t_{OS}, ali najčešće je ipak $t_{\mathrm{CPU}} \gg t_{OS}$
- U nastavku ćemo detaljnije promotriti t_{CPU}

PERFORMANSA: RADNA FREKVENCIJA PROCESORA

Uloga frekvencije signala takta procesora f_{CPU}:

- digitalni sklopovi izvode mikrooperacije u diskretnim periodima signala takta
- trajanje takta određeno najsporijom mikrooperacijom
- npr, $t_{\mu_{op}} = 250ps \Rightarrow f_{\mathrm{CPU}} = 4\,\mathrm{GHz}$
- u načelu, brži takt implicira veću performansu, ali svi znamo da veza nije jednostavna
  - računala rade na $2 - 3\,\mathrm{GHz}$ već 5 godina, ali performansa ipak raste oko 20% godišnje ($1{,}2^5 \approx 2{,}5$!)
  - $\Rightarrow$ danas računala naprave više u jednom periodu (ciklusu) signala takta nego prije 5 godina

PERFORMANSA: UTJECAJ RADNE FREKVENCIJE

Formalizirajmo vaeu između procesorskog vremena i radnog takta:

$$t_{\mathrm{CPU}} = n_{ciklusa} \cdot T_{\mathrm{takt}} = \frac{n_{ciklusa}}{f_{\mathrm{CPU}}}$$

Vidimo da će performansi pogodovati:

- smanjenje broja taktova pri izvođenju programa
- povećanje radne frekvencije

PERFORMANSA: UTJECAJ RADNE FREKVENCIJE

Formalizirajmo vaeu između procesorskog vremena i radnog takta:

$$t_{\mathrm{CPU}} = n_{ciklusa} \cdot T_{\mathrm{takt}} = \frac{n_{ciklusa}}{f_{\mathrm{CPU}}}$$

Vidimo da će performansi pogodovati:

- smanjenje broja taktova pri izvođenju programa
- povećanje radne frekvencije

Arhitekti obično traže optimalan kompromis između radne frekvencije f i broja ciklusa n_c ...

PERFORMANSA: UTJECAJ RADNE FREKVENCIJE

Formalizirajmo vaeu između procesorskog vremena i radnog takta:

$$t_{\mathrm{CPU}} = n_{ciklusa} \cdot T_{\mathrm{takt}} = \frac{n_{ciklusa}}{f_{\mathrm{CPU}}}$$

Vidimo da će performansi pogodovati:

- smanjenje broja taktova pri izvođenju programa
- povećanje radne frekvencije

Arhitekti obično traže optimalan kompromis između radne frekvencije f i broja ciklusa n_c ...

... osim ako se u igru ne umiješa marketinški odio :-)

PERFORMANSA: $n = t \times f$

Pretpostavimo da na tržištu postoji računalo A koje uz $f_A = 2\,\mathrm{GHz}$ ispitni program izvodi $t_A = 10\,\mathrm{s}$.

Pretpostavimo da nam je ambicija projektirati računalo B uz $t_B = 6\,\mathrm{s}$.

PERFORMANSA: $n = t \times f$

Pretpostavimo da na tržištu postoji računalo A koje uz $f_A = 2\,\mathrm{GHz}$ ispitni program izvodi $t_A = 10\,\mathrm{s}$.

Pretpostavimo da nam je ambicija projektirati računalo B uz $t_B = 6\,\mathrm{s}$.

Znamo da možemo postići znatno brži takt, ali po cijenu da n_{cB} bude jednak $1{,}2 \cdot n_{cA}$.

Kolika mora biti radna frekvencija računala B?

PERFORMANSA: $n = t \times f$

Pretpostavimo da na tržištu postoji računalo A koje uz $f_A = 2\,\mathrm{GHz}$ ispitni program izvodi $t_A = 10\,\mathrm{s}$.

Pretpostavimo da nam je ambicija projektirati računalo B uz $t_B = 6\,\mathrm{s}$.

Znamo da možemo postići znatno brži takt, ali po cijenu da n_{cB} bude jednak $1{,}2 \cdot n_{cA}$.

Kolika mora biti radna frekvencija računala B?

Rješenje:

- $n_{cA} = t_A \cdot f_A$
- $f_B = \frac{n_{cB}}{t_B} = 1{,}2 \cdot \frac{t_A}{t_B} \cdot f_A = 4\,\mathrm{GHz}$

PERFORMANSA: NC = NI X CPI

U jednadžbu performanse uvodimo broj instrukcija
(možemo mjeriti, za postojeće programe je konstantan).

CPI — broj ciklusa po instrukciji, engl. cycles per instruction
(veza između broja taktova n_c i broja instrukcija n_i):

$$n_c = n_i \cdot \mathrm{CPI}$$

PERFORMANSA: NC = NI X CPI

U jednadžbu performanse uvodimo broj instrukcija
(možemo mjeriti, za postojeće programe je konstantan).

CPI — broj ciklusa po instrukciji, engl. cycles per instruction
(veza između broja taktova n_c i broja instrukcija n_i):

$$n_c = n_i \cdot \mathrm{CPI}$$

Konačni oblik jednadžbe procesorske performanse:

$$t_{\mathrm{CPU}} = \frac{n_i \cdot \mathrm{CPI}}{f_{\mathrm{CPU}}} = n_i \cdot \mathrm{CPI} \cdot T_{\mathrm{takt}}$$

PERFORMANSA: NC = NI X CPI

U jednadžbu performanse uvodimo broj instrukcija
(možemo mjeriti, za postojeće programe je konstantan).

CPI — broj ciklusa po instrukciji, engl. cycles per instruction
(veza između broja taktova n_c i broja instrukcija n_i):

$$n_c = n_i \cdot \mathrm{CPI}$$

Konačni oblik jednadžbe procesorske performanse:

$$t_{\mathrm{CPU}} = \frac{n_i \cdot \mathrm{CPI}}{f_{\mathrm{CPU}}} = n_i \cdot \mathrm{CPI} \cdot T_{\mathrm{takt}}$$

Još jedan zapis jednadžbe:

$$t_{\mathrm{CPU}} = \mathrm{instrukcije} \cdot \frac{\mathrm{ciklusi}}{\mathrm{instrukcija}} \cdot \frac{\mathrm{sekunde}}{\mathrm{ciklus}}$$

PERFORMANSA: DETALJI

- n_i ... ukupan broj instrukcija koje se izvedu u okviru izvođenja programa, ovisi o:
  - **problemu**
  - inventivnosti prevoditelja
  - instrukcijskoj arhitekturi procesora

PERFORMANSA: DETALJI

- n_i ... ukupan broj instrukcija koje se izvedu u okviru izvođenja programa, ovisi o:
  - **problemu**
  - inventivnosti prevoditelja
  - instrukcijskoj arhitekturi procesora
- CPI ... prosječan broj ciklusa po instrukciji
  - ovisi o arhitekturi i **organizaciji** procesora
  - ako CPI nije konstantan (najčešće nije) uzimamo ponderiranu srednju vrijednost (težinske faktore određujemo empirijski)

PERFORMANSA: DETALJI

- n_i ... ukupan broj instrukcija koje se izvedu u okviru izvođenja programa, ovisi o:
  - **problemu**
  - inventivnosti prevoditelja
  - instrukcijskoj arhitekturi procesora
- CPI ... prosječan broj ciklusa po instrukciji
  - ovisi o arhitekturi i **organizaciji** procesora
  - ako CPI nije konstantan (najčešće nije) uzimamo ponderiranu srednju vrijednost (težinske faktore određujemo empirijski)
- f ... frekvencija radnog takta, ovisi o:
  - **tehnologiji** izvedbe integriranog sklopa
  - organizaciji procesora

Performansa: primjer

Procesor A: period takta: $250\,\mathrm{ps}$, $\mathrm{CPI}{=}2{,}0$

Procesor B: period takta: $500\,\mathrm{ps}$, $\mathrm{CPI}{=}1{,}2$

Procesori imaju istu instrukcijsku arhitekturu $(n_{iA} = n_{iB} = n_i)$

Zadatak: usporediti performansu dvaju procesora.

PERFORMANSA: PRIMJER

Procesor A: period takta: $250\,\mathrm{ps}$, $\mathrm{CPI}{=}2{,}0$

Procesor B: period takta: $500\,\mathrm{ps}$, $\mathrm{CPI}{=}1{,}2$

Procesori imaju istu instrukcijsku arhitekturu $(n_{iA} = n_{iB} = n_i)$

Zadatak: usporediti performansu dvaju procesora.

Rješenje:

- $t_A = n_i \cdot \mathrm{CPI}_A \cdot T_A$
- $t_B = n_i \cdot \mathrm{CPI}_B \cdot T_B$
- $p_A(B) = \frac{t_B}{t_A} = \frac{600}{500} = 1{,}2$

PERFORMANSA: PRIMJER

Procesor A: period takta: $250\,\mathrm{ps}$, $\mathrm{CPI}{=}2{,}0$

Procesor B: period takta: $500\,\mathrm{ps}$, $\mathrm{CPI}{=}1{,}2$

Procesori imaju istu instrukcijsku arhitekturu $(n_{iA} = n_{iB} = n_i)$

Zadatak: usporediti performansu dvaju procesora.

Rješenje:

- $t_A = n_i \cdot \mathrm{CPI}_A \cdot T_A$
- $t_B = n_i \cdot \mathrm{CPI}_B \cdot T_B$
- $p_A(B) = \frac{t_B}{t_A} = \frac{600}{500} = 1{,}2$

Procesor A na $4\,\mathrm{GHz}$ je 20% jači od procesora B na $2\,\mathrm{GHz}$

PERFORMANSA: CPI, DETALJNIJE

Kod modernih računala CPI ovisi o tipu instrukcije

(latencije ADD, MULSS, DIVSS: 1, 4, 14)

PERFORMANSA: CPI, DETALJNIJE

Kod modernih računala CPI ovisi o tipu instrukcije
(latencije ADD, MULSS, DIVSS: 1, 4, 14)

Kod modernih računala CPI ne ovisi samo o tipu instrukcije, nego i o dinamičkim uvjetima u trenutku izvođenja

- ipak, za svaki razred instrukcija r može se procijeniti očekivani CPI_r
- tipovi: zbrajanje/oduzimanje, memorijske, grananje, ...

PERFORMANSA: CPI, DETALJNIJE

Kod modernih računala CPI ovisi o tipu instrukcije
(latencije ADD, MULSS, DIVSS: 1, 4, 14)

Kod modernih računala CPI ne ovisi samo o tipu instrukcije, nego i o dinamičkim uvjetima u trenutku izvođenja

- ipak, za svaki razred instrukcija r može se procijeniti očekivani CPI_r
- tipovi: zbrajanje/oduzimanje, memorijske, grananje, ...

Nadalje, za relevantni skup programa, empirijskim metodama možemo utvrditi diskretnu **distribuciju** razreda instrukcija p_r (vrijedi $\sum_r p_r = 1$)

Tada CPI procesora možemo procijeniti kao:

$$\mathrm{CPI} = \sum_r p_r \mathrm{CPI}_r$$

PERFORMANSA: CPI, PRIMJER

Zadana su dva alternativna prijevoda (1, 2) istog programa. Prijevodi koriste instrukcije iz razreda A, B i C. Ocijeniti CPI za oba prijevoda.

razred r	A	B	C
CPI_r	1	2	3
n_{i1}	2	1	2
n_{i2}	4	1	1

PERFORMANSA: CPI, PRIMJER

Zadana su dva alternativna prijevoda (1, 2) istog programa. Prijevodi koriste instrukcije iz razreda A, B i C. Ocijeniti CPI za oba prijevoda.

razred r	A	B	C
CPI_r	1	2	3
n_{i1}	2	1	2
n_{i2}	4	1	1

$$\mathrm{CPI}_1 = \frac{2}{5} \cdot 1 + \frac{1}{5} \cdot 2 + \frac{2}{5} \cdot 3 = 2 \quad (n_{c1} = 10)$$

$$\mathrm{CPI}_2 = \frac{4}{6} \cdot 1 + \frac{1}{6} \cdot 2 + \frac{1}{6} \cdot 3 = 1{,}5 \quad (n_{c2} = 9)$$

Performansa: CPI, primjer 2

Pretpostavimo da smo napravili sljedeća mjerenja:

- učestalost i prosječni CPI operacija s pomičnim zarezom: 25%, 4
- prosječan CPI svih ostalih operacija: 1.33
- učestalost, CPI instrukcije FPSQR: 2%, 20

PERFORMANSA: CPI, PRIMJER 2

Pretpostavimo da smo napravili sljedeća mjerenja:

- učestalost i prosječni CPI operacija s pomičnim zarezom: 25%, 4
- prosječan CPI svih ostalih operacija: 1.33
- učestalost, CPI instrukcije FPSQR: 2%, 20

Usporediti sljedeće razvojne alternative (pretp. isti utjecaj na f):

1. smanjiti CPI instrukcije FPSQR na 2
2. smanjiti CPI **svih** FP instrukcija na $2{,}5$

PERFORMANSA: CPI, PRIMJER 2

Pretpostavimo da smo napravili sljedeća mjerenja:

- učestalost i prosječni CPI operacija s pomičnim zarezom: 25%, 4
- prosječan CPI svih ostalih operacija: 1.33
- učestalost, CPI instrukcije FPSQR: 2%, 20

Usporediti sljedeće razvojne alternative (pretp. isti utjecaj na f):

1. smanjiti CPI instrukcije FPSQR na 2
2. smanjiti CPI **svih** FP instrukcija na $2{,}5$

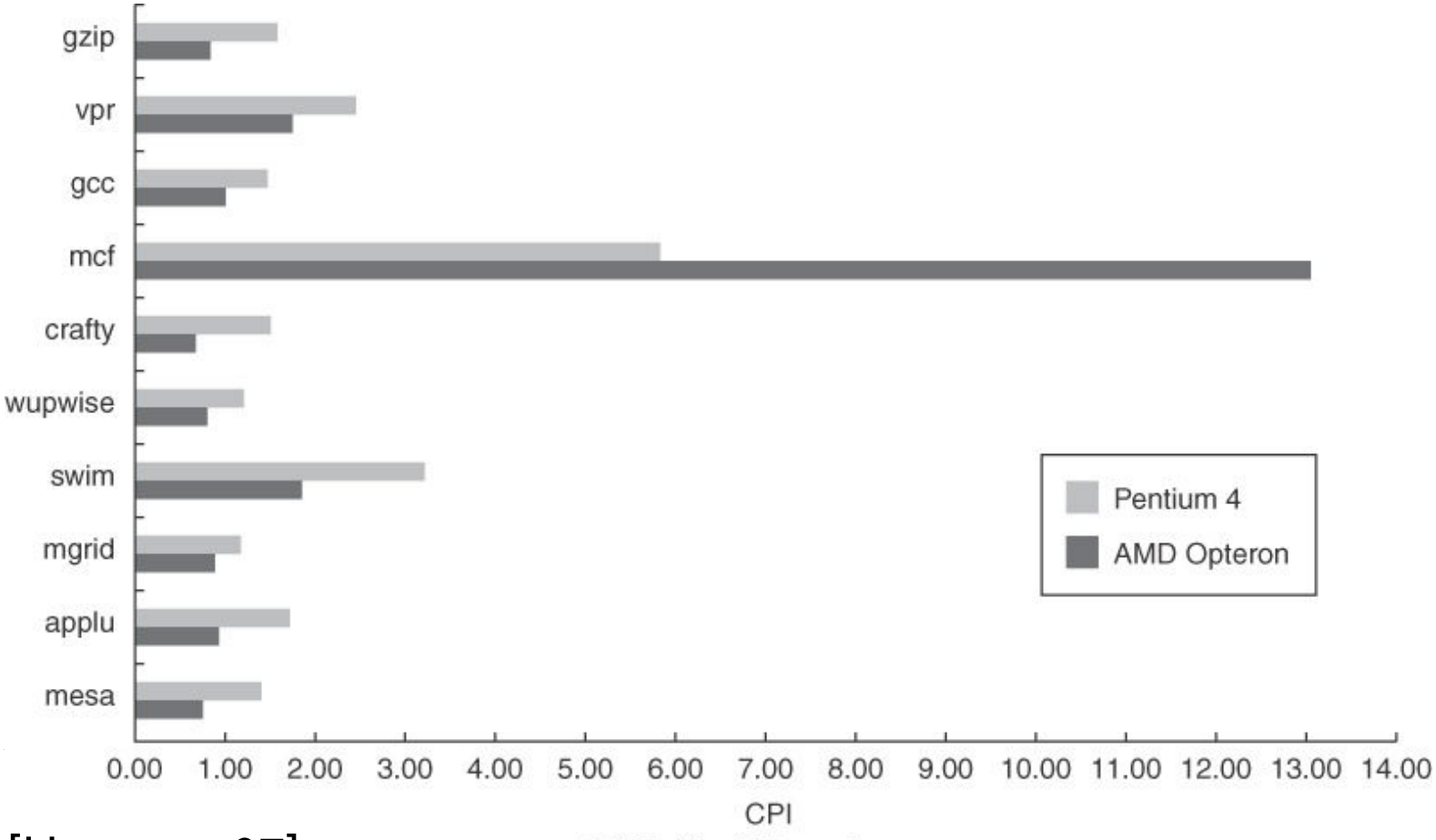
Rješenje:

1. primijetiti: n_i ostaje isti (novi CPU, stari programi!)
2. $\mathrm{CPI}_{orig} = 0{,}75 \cdot 1{,}33 + 0{,}25 \cdot 4 = 2$
3. $\mathrm{CPI}_1 = \mathrm{CPI}_{orig} - 2\% \cdot 20 + 2\% \cdot 2 = 1{,}64$
4. $\mathrm{CPI}_2 = \mathrm{CPI}_{orig} - 25\% \cdot 4 + 25\% \cdot 2{,}5 = 1{,}63$

PERFORMANSA: CPI U PRAKSI

CPI možemo mjeriti tijekom izvođenja reprezentativnih programa:

$$\mathrm{CPI} = \frac{t_{\mathrm{CPU}} \cdot f_{\mathrm{CPU}}}{n_i}$$



[Hennessy07]

PERFORMANSA: MEĐUSAŽETAK

- A je n puta brži od B ako:

$$n = P_A(B) = \frac{P_A}{P_B} = \frac{t_B}{t_A}$$

- U procesorski intenzivnim aplikacijama (bez U-I, VM, OS) vrijedi:

$$t_{\mathrm{CPU}} = n_i \cdot \mathrm{CPI} \cdot T_{\mathrm{takt}} = \text{broj instrukcija} \cdot \frac{\text{ciklusi}}{\text{instrukcija}} \cdot \frac{\text{sekunde}}{\text{ciklus}}$$

- Performansa ovisi o:
  - problemu (uglavnom preko n_i)
  - prevoditelju (n_i, CPI)
  - instrukcijskoj arhitekturi (n_i, CPI, f)
  - sklopovskoj organizaciji (CPI, f)
  - tehnologiji (f)

PERFORMANSA: EVALUACIJA

Kako evaluirati t_{CPU}^{-1}, $t_{\text{CPU}}^{-1}W^{-1}$, mrežu, baze, ...?

- trebaju nam referentni evaluacijski programi (engl. benchmark)
- evaluacijski programi mogu biti:
  - važni stvarni programi (gcc, perl, gzip),
  - sintetički programi koji simuliraju stvarna opterećenja (npr, dhrystone, whetstone, CoreMark...)
- kako bi se povećala stabilnost, obično se koriste evaluacijske kolekcije (engl. benchmark suites)
- Ozbiljnije evaluacijske kolekcije održavaju specijalizirane tvrtke:
  - Standard Performance Evaluation Corporation (SPEC)
  - Transaction Processing Performance Council (TPC)
  - Embedded Microprocessor Benchmark Consortium (EEMBC)
  - ...

PERFORMANSA: SPEC

Standard Performance Evaluation Corporation (`http://www.spec.org`):

- održava i razvija evaluacijske programe za računala opće namjene
- procesori, mrežni servisi, performansa/snaga, Java, itd.
- ocjene (SPECmark) odgovaraju **relativnoj performansi**
- lako se pokaže da odabir referentnog računala nije važan

PERFORMANSA: SPEC

Standard Performance Evaluation Corporation (`http://www.spec.org`):

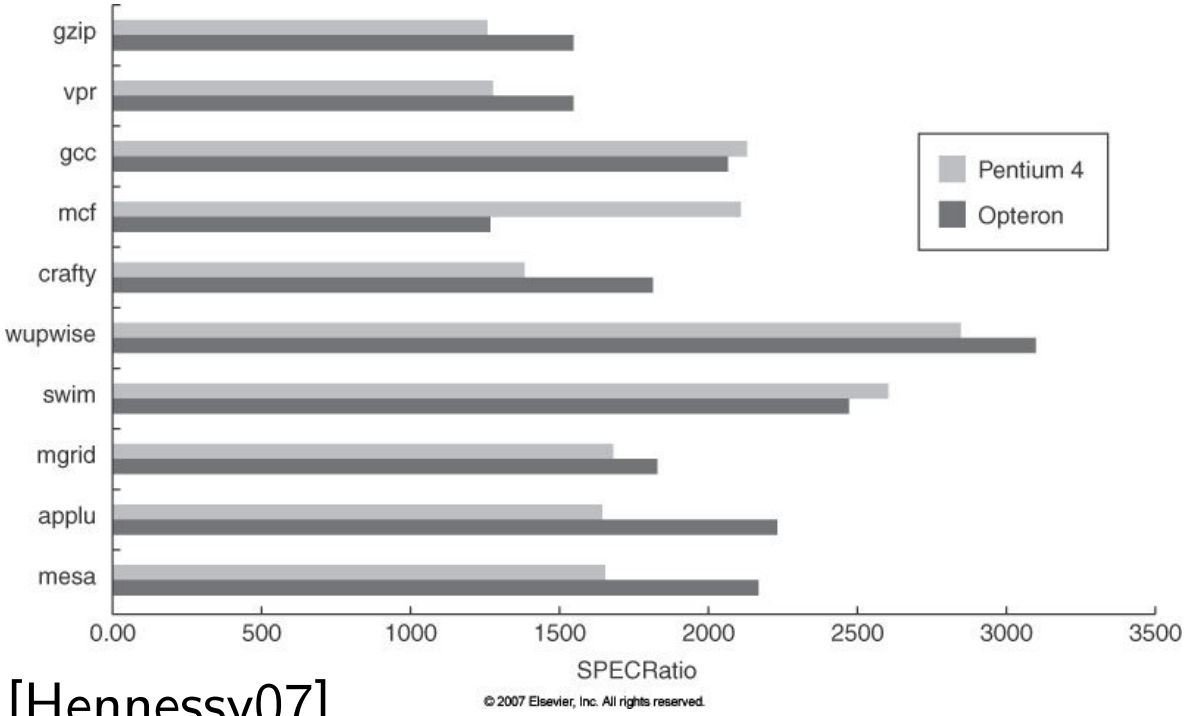
- održava i razvija evaluacijske programe za računala opće namjene
- procesori, mrežni servisi, performansa/snaga, Java, itd.
- ocjene (SPECmark) odgovaraju **relativnoj performansi**
- lako se pokaže da odabir referentnog računala nije važan

Procesorske kolekcije ciljaju cjelobrojnu (CINT) i FP performansu (CFP)

- mjeri se **brzina** (CINT2006) i propusnost (CINT2006_rate)
  - propusnost odgovara trajanju usporednih instanci istog programa
- za svaki test dodjeljuju se dvije ocjene:
  - osnovna (base) $\Rightarrow$ iste opcije prevoditelja za cijelu kolekciju
  - vršna (peak) $\Rightarrow$ opcije prilagođene pojedinačnim testovima

PERFORMANSA: ZBIRNA OCJENA

Pokazuje se da je rasipanje rezultata na pojedinim testovima veliko:



[Hennessy07]

$\Rightarrow$ ukupnu ocjenu određujemo kao geometrijsku sredinu pojedinačnih (aritmetička sredina omjera nema smisla):

$$P_{X,\mathrm{SPEC}}(R) = \sqrt[n]{\prod_i P_{X,i}(R)}$$

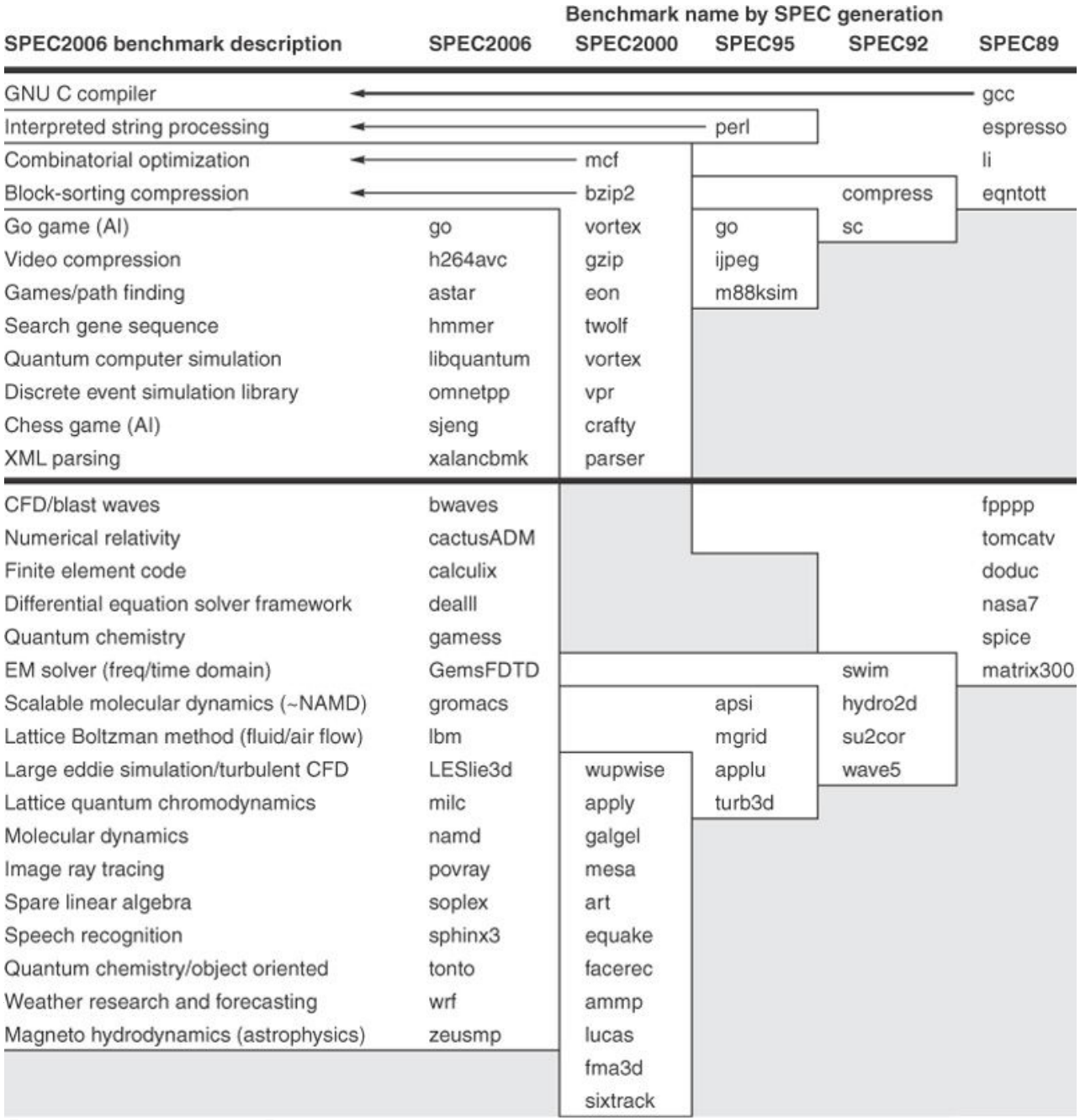
PERFORMANSA: SPEC 2006, PRIMJER

Koji je procesor bolji odabir za specifične ((peak)) aplikacije?

CPU (SPECmark peak)	cijena	CINT	CFP	CINTr	CFPr
Intel Core 2 Quad Q8400	1300 kn	22	20	59	41
AMD Phenom II X4 955	1400 kn	19	19	53	43

U ovom trenutku Intel nudi više performanse za novac.

PERFORMANSA: SPEC-OVI PROGRAMI



SPEC2006 benchmark description	Benchmark name by SPEC generation: SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
GNU C compiler					gcc
Interpreted string processing			perl		espresso
Combinatorial optimization		mcf			li
Block-sorting compression		bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	ijpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealII				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ammp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

[Hennessy07]

PERFORMANSA: SPEC-OVI PROGRAMI

Prethodna slika prikazuje evoluciju kolekcija SPEC CINT i SPEC CFP

Referentni programi se moraju odabrati tako da kolekcija odražava **tipična** stvarna opterećenja

Dodatno, proizvođači se **prilagođavaju** referentnim programima, pa je programe često potrebno mijenjati i prije nego što zastare

Razdioba programskih jezika u CINT 2006: (C: 9; C++: 3)

Razdioba programskih jezika u CFP 2006: (Fortran: 6; C++: 4; C: 3; C/Fortran: 4)

Performansa: pouke

Jednadžba procesorske performanse:

$$t_{\mathrm{CPU}} = n_i \cdot \mathrm{CPI} \cdot T_{\mathrm{takt}}$$

Kako povećati performansu u okvirima postojeće tehnologije?

- usredotočiti se na najčešći slučaj ($\mathrm{CPI} \cdot T_{\mathrm{takt}} \downarrow$):
  npr, prilagoditi put podataka najvažnijim mikroperacijama,
- iskoristiti paralelizam ($\mathrm{CPI} \downarrow$):
  npr, predvidjeti dobro popunjenu protočnu strukturu
- iskoristiti lokalnost pristupa memoriji ($T_{\mathrm{takt}} \downarrow$):
  npr, predvidjeti registarske skupove i priručne memorije

Ključna tri načela pri projektiranju sklopova s kompetitivnom performansom!

PERFORMANSA: NAJČEŠĆI SLUČAJ

Načelo: kad se pojavi potreba za kompromisom, favoriziraj česti slučaj

- npr, pribavljanje instrukcije češće od množenja
  $\Rightarrow$ optimirati fazu pribavljanja!
- npr, množenje češće od dijeljenja (pristup elementu matrice)
  $\Rightarrow$ množenju alocirati više tranzistora!

PERFORMANSA: NAJČEŠĆI SLUČAJ

Načelo: kad se pojavi potreba za kompromisom, favoriziraj česti slučaj

- npr, pribavljanje instrukcije češće od množenja
  $\Rightarrow$ optimirati fazu pribavljanja!
- npr, množenje češće od dijeljenja (pristup elementu matrice)
  $\Rightarrow$ množenju alocirati više tranzistora!

Kako se favoriziranje čestog slučaja preslikava u ukupnu performansu?

PERFORMANSA: NAJČEŠĆI SLUČAJ

Načelo: kad se pojavi potreba za kompromisom, favoriziraj česti slučaj

- npr, pribavljanje instrukcije češće od množenja
  $\Rightarrow$ optimirati fazu pribavljanja!
- npr, množenje češće od dijeljenja (pristup elementu matrice)
  $\Rightarrow$ množenju alocirati više tranzistora!

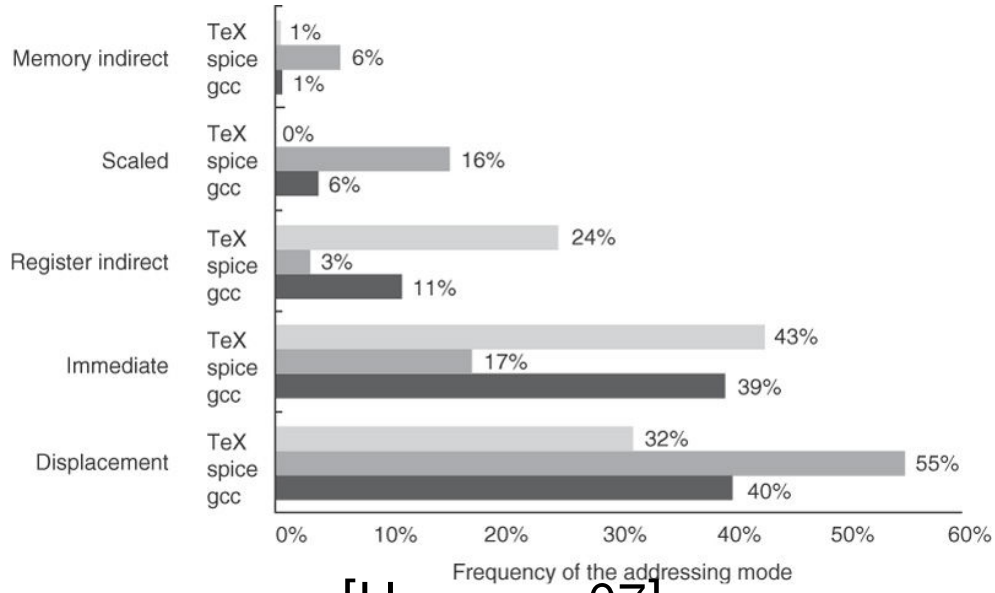
Kako se favoriziranje čestog slučaja preslikava u ukupnu performansu?

$\rightarrow$ Amdahlov zakon!

PERFORMANSA: NAJČEŠĆI I NAJJEDNOSTAVNIJI

Česti slučaj **obično** ujedno i jednostavniji pa se može implementirati brže

- pokazuje se da se složeni načini adresiranja rijetko koriste
  $\Rightarrow$ treba optimirati jednostavne načine adresiranja
- to će unekoliko usporiti slučajeve u kojima se složeni načini adresiranja *mogu* koristiti, ali ukupna performansa će se povećati!
- usputno, registarsko indirektno, te registarsko indirektno adresiranje s pomakom su na VAX-u tipično zastupljeni s preko 75% (slika!)



[Hennessy07]

PERFORMANSA: PARALELIZAM

Načelo: povećati performansu usporednom obradom

- npr, povećati propusnost poslužitelja povećanjem broja procesora ili diskova
- npr, povećati memorijsku propusnost prepletenim spremanjem podataka na više sklopova
- protočno i superskalarno izvršavanje instrukcija

PERFORMANSA: PARALELIZAM

Načelo: povećati performansu usporednom obradom

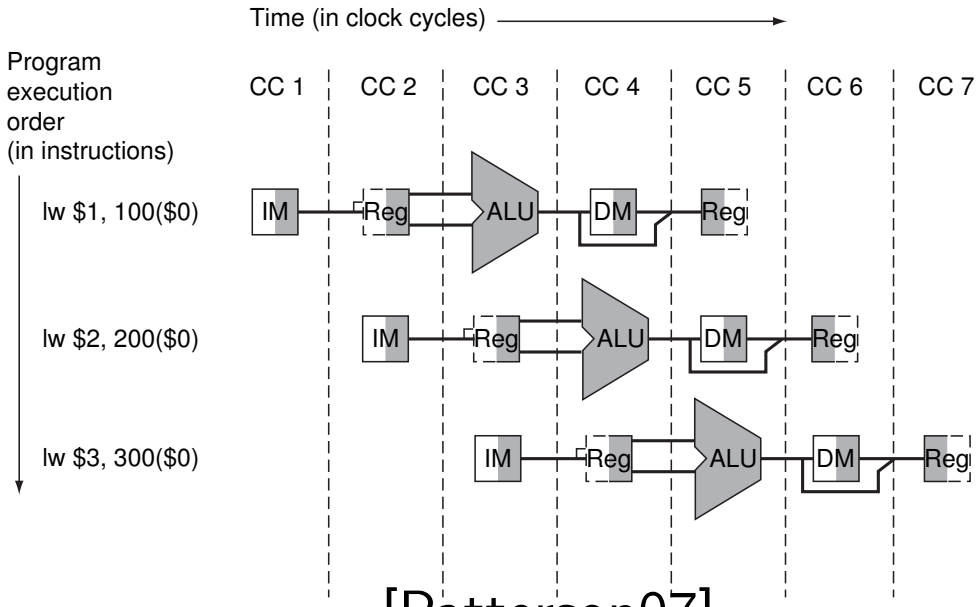
- npr, povećati propusnost poslužitelja povećanjem broja procesora ili diskova
- npr, povećati memorijsku propusnost prepletenim spremanjem podataka na više sklopova
- protočno i superskalarno izvršavanje instrukcija

Protočnost je posebno efikasan način povećanja performanse

- etape susjednih instrukcija najčešće se mogu izvršavati usporedno (ponovo se fokusiramo na najčešći slučaj)
- $\mathrm{CPI} \downarrow \times 4$

PERFORMANSA: PARALELIZAM, PRIMJER

Protočno izvođenje instrukcija u arhitekturi MIPS:



[Patterson07]

- postiže se $\mathrm{CPI} \approx 1$, iako latencija svake instrukcije iznosi 5 ciklusa,
- pokazuje se da je ideja ograničena međuovisnostima među instrukcijama (hazardima)
- iako postoje, dublje protočne strukture najčešće nisu opravdane (npr, Pentium 4 Prescott - 30 segmenata)

PERFORMANSA: LOKALNOST

Načelo: iskoristiti lokalnost pristupa memoriji

- još jedan važan empirijski potvrđen česti slučaj
- veliki registarski skupovi i priručne memorije:
  $\longrightarrow$ lijek za Von Neumannovo usko grlo!
- postoji prostorna i vremenska lokalnost

PERFORMANSA: LOKALNOST

Načelo: iskoristiti lokalnost pristupa memoriji

- još jedan važan empirijski potvrđen česti slučaj
- veliki registarski skupovi i priručne memorije:
  $\longrightarrow$ lijek za Von Neumannovo usko grlo!
- postoji prostorna i vremenska lokalnost

Usporedimo vremena potrebna za izvođenje tipičnih μoperacija:

- istovremeno čitanje dva registra te upisivanje u treći $(1\,\Delta\mathrm{T})$
- pristup priručnoj memoriji L1 $(1\,\Delta\mathrm{T})$, L2 $(5\,\Delta\mathrm{T})$ i L3 $(25\,\Delta\mathrm{T})$
- pristup glavnoj memoriji $(100\,\Delta\mathrm{T})$
- pristup disku $(10^6\,\Delta\mathrm{T})$

PERFORMANSA: SAŽETAK

Jednadžba procesorske performanse:

$$t_{\mathrm{CPU}} = n_i \cdot \mathrm{CPI} \cdot T_{\mathrm{takt}}$$

Procesorsku performansu mjerimo kolekcijama pažljivo odabranih ispitnih programa

Moderna načela za ostvarivanje visoke performanse:

- usredotočiti se na najčešći slučaj
- iskoristiti paralelizam
- iskoristiti načelo lokalnosti

Performansa: literatura

1. Computer Organization and Design, 4th ed, David Patterson and John Hennessy, Morgan Kaufmann
2. Computer Architecture: A Quantitative Approach, 4th ed, John Hennessy and David Patterson Morgan Kaufmann
3. Arhitektura računala CISC i RISC, Slobodan Ribarić, Školska knjiga 1996