הערכת ביצועים

ארגון המחשב ושפת סף

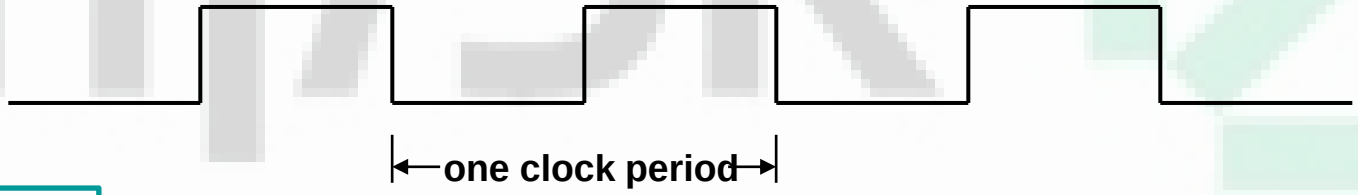
מרצה: ***רועי אש***

אפקה
המכללה האקדמית
להנדסה בתל אביב

Review: Machine Clock Rate

- **C**lock **R**ate (MHz, GHz) is inverse of clock cycle time (clock period)

CCT = 1 / CR



K	$= 10^3$	קילו
M	$= 10^6$	מגה
G	$= 10^9$	גיגה
T	$= 10^{12}$	טרה

ms	$= 10^{-3}$	מילי
µs	$= 10^{-6}$	מיקרו
ns	$= 10^{-9}$	ננו
ps	$= 10^{-12}$	פיקו

10 nsec clock cycle => 100 MHz clock rate
5 nsec clock cycle => 200 MHz clock rate
2 nsec clock cycle => 500 MHz clock rate
1 nsec clock cycle => 1 GHz clock rate
500 psec clock cycle => 2 GHz clock rate
250 psec clock cycle => 4 GHz clock rate
200 psec clock cycle => 5 GHz clock rate

חישוב זמן CPU

- **CPU Time - time required to execute a program**

  (response time ,execution time, elapsed time, real time, run time, latency)

CPU Time = IC × CPI × CCT **[sec/program]**

כאשר:

IC (Instruction Count) – מספר הפקודות בתוכנית

CPI (Clock Per Instruction) – מספר המחזורים שנדרש לפקודה אחת (ממוצע)

CCT (Clock Cycle Time) – הזמן שנמשך מחזור שעון אחד בשניות

לחילופין, כאשר:

$$CPUtime = \frac{IC \cdot CPI}{CR}$$

CC (Clock Cycle) = **IC*CPI** – מספר מחזורי השעון של המעבד בתוכנית

היעד המהותי – צמצום זמן CPU

- Our goal: minimize CPU Time
  - Minimize clock cycle time : more GHz (process, circuit, μArch)
  - Minimize CPI: μArch (e.g.: more execution units)
  - Minimize IC: architecture (e.g.: MMX™ technology)

- Throughput – the total amount of work done in a given time
  - Important to data center managers
- Decreasing response time almost always improves throughput

CPI and the ISA

- Comparing implementations of an ISA
  - CPI provides one way of comparing different implementations of the same ISA
  - The number of instructions executed for a given program is always the same

גורם ההאצה - Speedup

"F(ast) is *n* times faster than S(low)" means…

$$N = \text{Speedup} = \frac{\text{CPU\_time}(S)}{\text{CPU\_time}(F)}$$

תרגול 1

נתונות שתי מכונות M1 ו- M2. בשתיהן אותו סט פקודות.
ניתן לחלק את הפקודות לקבוצות A B C D
להלן ה- CPI של ארבעת סוגי הפקודות בשתי המכונות:

	M1	M2
A	1	2
B	2	2
C	3	4
D	4	4

ב- M1 תדר שעון 75Mhz
ב- M2 תדר שעון 50Mhz

א. peek performance (ביצועי שיא) מוגדרים כביצועים הטובים ביותר שמחשב יכול "להפגין". מאילו סוגי פקודות ניתן לקבל קוד שייתן ביצועי שיא ב M1 וב- M2?
ב. מהו ה CPI ל- M1 ול-M2 בהנחה ששכיחות סוגי הפקודות הינה 25% לכל סוג?
ג. איזו מכונה מהירה יותר תחת ההנחה של סעיף ב ?

תרגול 1: תשובה

א. כדי "להפגין" ביצועים כמה שיותר טובים, נשתמש רק בפקודות שה-CPI שלהם הכי קטן.

– ב-M1, אלו פקודות מסוג A בלבד.

– ה-M2, אלו פקודות מסוג A או B.

	M1	M2
A	1	2
B	2	2
C	3	4
D	4	4

– ברור שמדד ה-peak performance מאוד מטעה – בתוכנית "אמיתית" אנו נאלצים להשתמש בפקודות מסוגים שונים

תרגול 1: תשובה (המשך)

ב. ה-CPI למעשה יהיה ממוצע יחסי של הפקודות, עפ"י "משקלן" בקוד תוכנית טיפוסית, כפי שנתון.

	M1	M2
A	1	2
B	2	2
C	3	4
D	4	4

$$CPI = \sum_{i=1}^{T} w_i CPI_i$$

מכיוון שלכל סוג אותו משקל יחסי, נחשב ממוצע "רגיל" בין הסוגים:

$$M_1: \quad CPI_1 = \frac{CPI_A + CPI_B + CPI_C + CPI_D}{4} = \frac{1+2+3+4}{4} = \frac{10}{4} = 2.5$$

$$M_2: \quad CPI_2 = \frac{CPI_A + CPI_B + CPI_C + CPI_D}{4} = \frac{2+2+4+4}{4} = \frac{12}{4} = 3$$

תרגול 1: תשובה (המשך)

ג. פשוט נחשב את זמן ה-CPU לכל מכונה: $CPUtime = \frac{IC \cdot CPI}{CR}$

$$CPUtime_1 = \frac{IC \cdot 2.5}{75 \cdot 10^6} = \frac{1}{3} \cdot 10^{-7} IC$$

$$CPUtime_2 = \frac{IC \cdot 3}{50 \cdot 10^6} = 0.6 \cdot 10^{-7} IC$$

- $CPUtime_1 < CPUtime_2$
- לכן ברור שהמכונה הראשונה יותר מהירה.

האם קוד קטן הוא גם מהיר (smaller is faster)?

אשר או הכחש.

נמק תשובתך.

- ממש לא.
  - אמנם יש פחות פקודות (IC)
  - אולם לא התייחסנו לכמות מחזורי השעון שצורכת כל פקודה (CPI)
  - וגם (אם רלוונטי) למשך מחזור שעון אחד בשניות (CCT).
- לדוגמא: במכונת CISC יהיו באמת מעט שורות קוד, אך ביצוע כל שורת קוד כזאת ידרוש הרבה יותר מחזורי שעון מאשר קוד שמבצע את אותה הפעולה במכונת RISC.

Determinates of CPU Performance

CPU time = Instruction_count x CPI x clock_cycle_time

	Instruction Count	Clock Per Instruction	Clock Cycle Time
Algorithm	X	X	
Programming language	X	X	
Compiler	X	X	
ISA	X	X	X
Processor Organization		X	X
Technology			X

תרגול 2

Op	Freq	CPI_i	$\omega_i \times CPI_i$
ALU	50%	1	
Load	20%	5	
Store	10%	3	
Branch	20%	2	
			$\Sigma =$

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?
- How does this compare with using branch prediction to shave a cycle off the branch time?
- What if two ALU instructions could be executed at once?

תרגול 2: תשובה

Op	Freq	CPI_i		ω_i x CPI_i		
ALU	50%	1	.5	.5	.5	.25
Load	20%	5	1.0	.4	1.0	1.0
Store	10%	3	.3	.3	.3	.3
Branch	20%	2	.4	.4	.2	.4
			2.2	Σ = 1.6	2.0	1.95

- How much faster would the machine be if a better data cache reduced the average load time to 2 cycles?

  CPU time new = 1.6 x IC x CCT so 2.2/1.6 means 37.5% faster

- How does this compare with using branch prediction to shave a cycle off the branch time?

  CPU time new = 2.0 x IC x CCT so 2.2/2.0 means 10% faster

- What if two ALU instructions could be executed at once?

  CPU time new = 1.95 x IC x CCT so 2.2/1.95 means 12.8% faster

מדדים "אובייקטיביים"

MIPS = Millions of Instructions Per Second

כמה מיליוני פעולות מבצע המעבד בשנייה

$$MIPS = \frac{\text{Instruction count}_{(instructions)}}{\text{CPU time}_{(sec)} * 10^6} = \frac{clock\ rate_{(cycle/sec)}}{CPI_{(cycle/instruction)} * 10^6}$$

בעיה: מדד MIPS תלוי בסט הפקודות של המכונה ובהרכב תכנית הבדיקה. אם במעבד מסוים יש פקודות אסמבלר מורכבות יותר, שלוקח הרבה זמן לבצע אותן, אבל עקב כך צריך להשתמש בפחות פקודות מאשר במעבדים אחרים כדי לבצע את אותם דברים, אז מדד ה- MIPS שיתקבל עבור אותו מחשב יהיה נמוך, ולא בהכרח בצדק.

חוק אמדל – Amdahl's Law

- ניקח מכונה כלשהי שזמן הריצה של תכנית P עליה הוא $ExTime_{old}$.
- נניח שאנו משפרים את המעבד כך שחלק Fraction מזמן ריצת התכנית ירוץ פי Speedup יותר מהר בזכות השיפור.
- השאלה היא מה יהא זמן הריצה כעת?

$$ExTime_{new} = ExTime_{old} \times \left[(1 - Fraction) + \frac{Fraction}{Speedup} \right]$$

חוק אמדל

- פועל יוצא הוא שחישוב ההאצה הכללית שהשגנו ניתן לחישוב באופן הבא:

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

תרגול 3

ידועים לנו שני חסרונות במעבד מסוים:

א. זמן הביצוע של Branch הינו ארוך מדי.

ב. זמן הטיפול ב overflow אריתמטי גם הוא ארוך.

- ביכולתנו לשפר את זמן הביצוע ב- א' ב20% ואילו ב- ב' פי 8.
- ידוע שתדירות פקודות Branch כ 15% ותדירות overflow הינה 0.001%.
- באיזה שיפור כדאי לבחור בהנחה שמגבלות התקציב אינן מתירות את שניהם?

"לשפר את זמן הביצוע ב-א' ב-20%"

- שתי דרכים להבין זאת:
  - זמן הביצוע החדש של פקודת branch הוא 80% מזמן הביצוע הישן.

$$Speedup_{branch} = 1 / 0.8 = 1.25 = 125\%$$

  - מדד ה-Speedup הוא בעצם חלוקת הזמן הישן בזמן החדש. במילים אחרות, Speedup לוקח את הזמן החדש להיות ה-100%, ולכן הזמן הישן של פקודות ה-branch הוא 120% מהחדש.

$$\mathbf{Speedup_{branch} = 1.2 / 1.0 = 1.2 = 120\%}$$

- אנו נשתמש בקורס בפירוש **השני**.

תרגול 3: תשובה

- נחשב את זמן ה-CPU של כל אפשרות:

$$CPUtime_1 = IC \cdot 0.85 \cdot CPI \cdot CCT + IC \cdot 0.15 \cdot \frac{1}{1.2} \cdot CPI \cdot CCT$$

$$= IC \cdot (0.85 + 0.15 \cdot \frac{1}{1.2}) \cdot CPI \cdot CCT = 0.975 \cdot IC \cdot CPI \cdot CCT$$

$$= 0.975 \cdot CPUtime_{old}$$

$$CPUtime_2 = IC \cdot 0.99999 \cdot CPI \cdot CCT + IC \cdot 0.00001 \cdot \frac{1}{8} \cdot CPI \cdot CCT$$

$$= IC \cdot (0.99999 + 0.00001 \cdot 0.125) \cdot CPI \cdot CCT$$

$$= 0.99999125 \cdot IC \cdot CPI \cdot CCT$$

$$= 0.99999125 \cdot CPUtime_{old}$$

תרגול 3: תשובה (המשך)

- ההצעה הראשונה נותנת שיפור של 2.5%.
- ההצעה השנייה נותנת שיפור מזערי של 0.00875%.
- לכן ההצעה הראשונה הרבה יותר טובה.
- בצורה מקבילה ניתן לפתור זאת ע"י Amdahl's Law, שמחשב את אחוז השיפור בצורה מאוד דומה:

$$Speedup = \frac{1}{(1 - F_{enhanced}) + \frac{F_{enhanced}}{Speedup_{enhanced}}}$$

Amdahl's Law

מלכודת נפוצה: ציפייה שהשיפור של פרמטר אחד בביצועי המחשב ישפר ביצועים בסדר גודל יחסי לגודל השיפור.

המסקנה

יש לשפר את המקרה השכיח

How to compare between different systems?

אפקה
המכללה האקדמית
להנדסה בתל אביב

Benchmarks – Programs for Evaluating Processor Performance

- Toy Benchmarks
  - 10-100 line programs
  - e.g.: puzzle, quicksort
- Synthetic Benchmarks
  - Attempt to match average frequencies of real workloads
  - e.g., Winstone, Dhrystone
- Real programs
  - e.g., gcc, spice
- SPEC: System Performance Evaluation Cooperative

  SPEC CPU2006 which include:
  - 12 integer benchmarks (Cint2006)
  - 17 floating point benchmarks (CFP2006)

workload

Comparing and Summarizing Performance

- How do we summarize the performance for benchmark set with a single number?
  - The average of execution times that is directly proportional to total execution time is the arithmetic mean (AM)**

$$AM = 1/n \sum_{i=1}^{n} Time_i$$

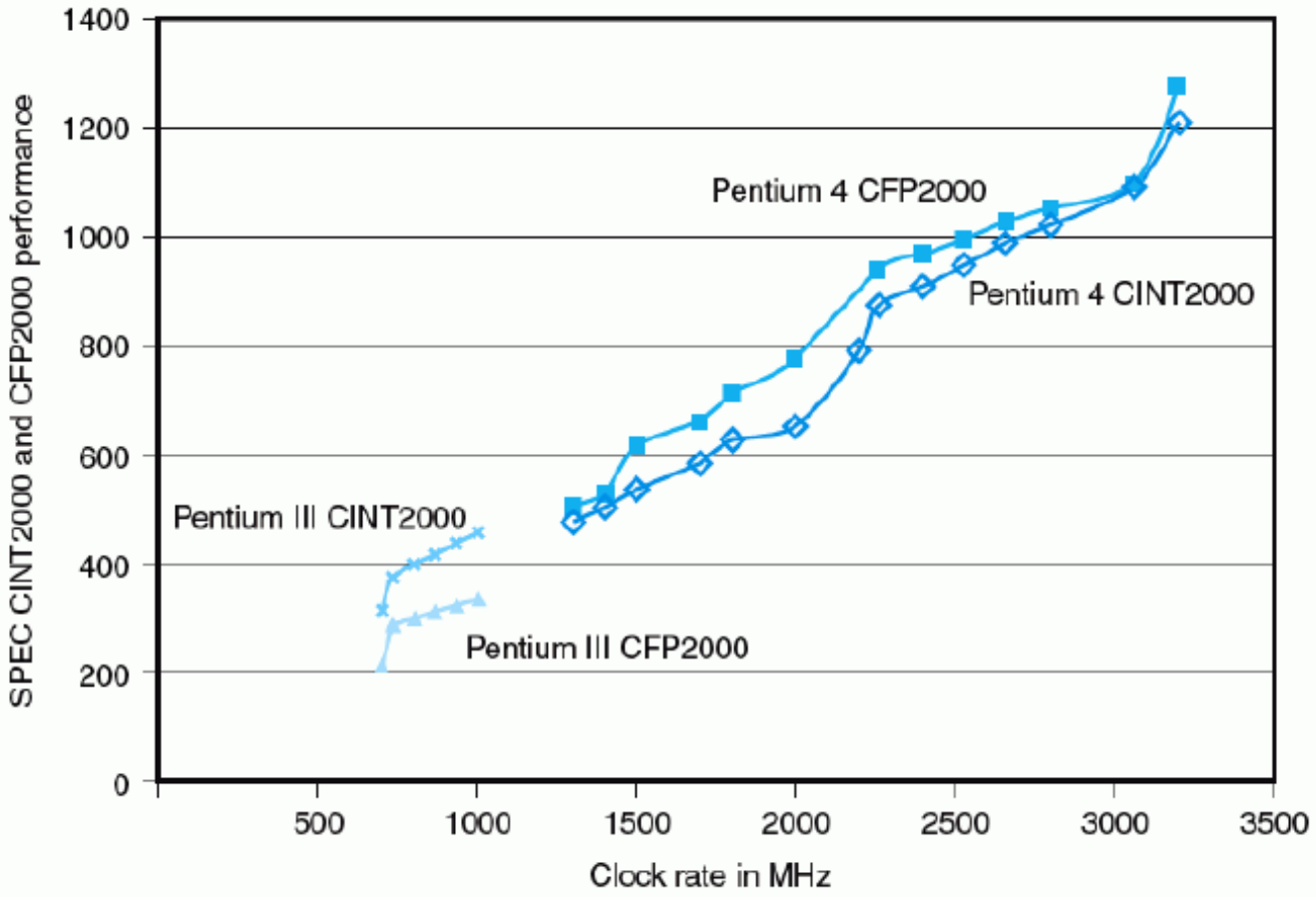
  - Where $Time_i$ is the execution time for the ith program of a total of n programs in the workload
  - A smaller mean indicates a smaller average execution time and thus improved performance

- Guiding principle in reporting performance measurements is reproducibility:
  list everything another experimenter would need to duplicate the experiment: version of the operating system, compiler settings, input set used, specific computer configuration (clock rate, cache sizes and speed, memory size and speed, etc.)
- ** Geometric mean

SPEC Benchmarks www.spec.org

Integer benchmarks		FP benchmarks	
gzip	compression	wupwise	Quantum chromodynamics
vpr	FPGA place & route	swim	Shallow water model
gcc	GNU C compiler	mgrid	Multigrid solver in 3D fields
mcf	Combinatorial optimization	applu	Parabolic/elliptic pde
crafty	Chess program	mesa	3D graphics library
parser	Word processing program	galgel	Computational fluid dynamics
eon	Computer visualization	art	Image recognition (NN)
perlbmk	perl application	equake	Seismic wave propagation simulation
gap	Group theory interpreter	facerec	Facial image recognition
vortex	Object oriented database	ammp	Computational chemistry
bzip2	compression	lucas	Primality testing
twolf	Circuit place & route	fma3d	Crash simulation fem
		sixtrack	Nuclear physics accel
		apsi	Pollutant distribution

Example SPEC Ratings



Other Performance Metrics

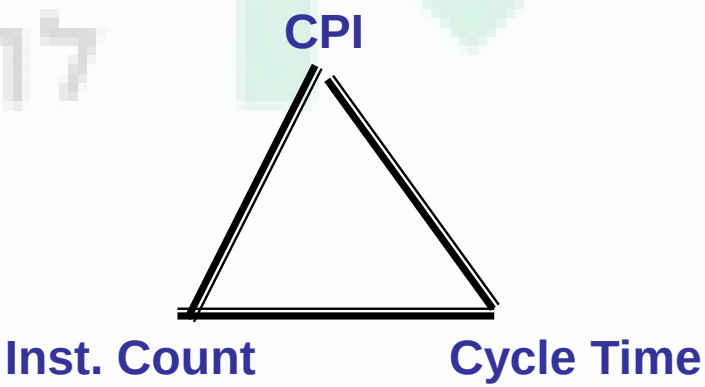
- Power consumption – especially in the embedded market where battery life is important (and passive cooling)
  - For power-limited applications, the most important metric is energy efficiency

Summary: Evaluating ISAs

- Design-time metrics:
  - Can it be implemented, in how long, at what cost?
  - Can it be programmed? Ease of compilation?
- Static Metrics:
  - How many bytes does the program occupy in memory?
- Dynamic Metrics:
  - How many instructions are executed? How many bytes does the processor fetch to execute the program?
  - How many clocks are required per instruction?
  - How "lean" a clock is practical?

Best Metric: Time to execute the program!

depends on the instructions set, the processor organization, and compilation techniques.



תרגול 4

- נתונים שני מעבדים שונים, P1 ו-P2, עם אותו אוסף פקודות. ניתן לחלק את הפקודות ל-4 קבוצות שונות: A, B, C ו-D.
- ל-P1 יש קצב שעון של GHz1.5 , ול-P2 קצב שעון של GHz 2.
- המספר הממוצע של מחזורי שעון לכל קבוצת פקודות עבור P1 ו-P2 הוא כדלקמן:

קבוצה	CPI על P1	CPI על P2	שכיחות בקוד
A	1	2	10%
B	2	2	20%
C	3	2	50%
D	4	2	20%

- בהינתן תכנית בת 10^6 פקודות, מצאו:
  א. מהו ה-CPI הממוצע לכל מעבד?
  ב. כמה מחזורי שעון נחוצים בשני המקרים?
  ג. איזה מעבד הוא יותר מהיר?

תרגול 4: תשובה

א. נחשב תחילה את - CPI הממוצע על כל מעבד

P1:

$CPI(P1) = 0.1*1[cc/ins] + 0.2*2[cc/ins] + 0.5*3[cc/ins] + 0.2*4[cc/ins] = 2.8\ [cc/ins]$

P2:

$CPI(P2) = 0.1*2[cc/ins] + 0.2*2[cc/ins] + 0.5*2[cc/ins] + 0.2*2[cc/ins] = 2\ [cc/ins]$

ב. נכפיל במספר הפקודות (IC) לקבל את מספר המחזורים בתכנית

$IC*CPI(p1) = 10^6[Ins/P]*2.8\ [Cc/ins] = 2.8*10^6[Cc/P]$

$IC*CPI(p2) = 10^6[Ins/P]*2\ [Cc/ins] = 2\ *10^6[Cc/P]$

ג. כדי למצוא את זמן הריצה נחלק בקצב השעון:

P1: $2.8*10^6[Cc/P]/1.5*10^9[Cc/sec] = 1.867*10^{-3}[sec/P]$

P2: $2\ *10^6[Cc/P]/2\ *10^9[Cc/sec] = 1*10^{-3}[sec/P]$

- **אנו רואים שהזמן של P2 יותר קצר ולכן הוא יותר מהיר**
- **מדד ההאצה הוא : 1.867**

תרגול 5

- נתון מעבד מסוים עם מימוש של אוסף פקודות. הפקודות מתחלקות ל-4 סוגים: A, B, C ו-D. קצב השעון במימוש זה הוא 1.5Ghz
- ה-CPI של כל סוג הוא:
  - A: 1 B: 2 C: 3 D: 4
- נתונה תכנית עם בה מורצות 10^8 פקודות המחולקת לסוגים הנ"ל לפי המשקלים הבאים:
  - A: 10% B: 20% C: 50% D: 20%

א) כמה זמן ייקח למעבד לבצע את התכנית?

ב) במידה והצלחנו לשפר את האלגוריתם של התכנית כך שיורצו רק $7x10^7$ פקודות אבל משקלי הפקודות שונו ל:
  - A: 10% B: 10% C: 30% D: 50%
  - האם שיפור האלגוריתם שיפר את זמן הריצה ? ואם כן מה מדד ההאצה?

תרגול 5: תשובה

א) **נחשב זמן ריצה:**

CPI = 0.1*1 + 0.2*2 + 0.5*3 + 0.2*4 = 2.8 [Cc/Inc]

CPU_{time} = IC*CPI/CR = 10^8[Ins/P]*2.8[Cc/Ins] / ($1.5*10^9$)[Cc/Sec]

= **<u>0.18667</u>** [sec/P)

ב) **עבור שיפור האלגוריתם נקבל:**

Cpi = 0.1*1 + 0.1*2 + 0.3*3 + 0.5*4 = 3.2 [Cc/Inc]

CPU_{time} = IC*CPI/CR = $7x10^7$[Ins/P]*3.2[Cc/Ins] / ($1.5*10^9$)[Cc/Sec]

= **<u>0.14933</u> [sec/P)**

כלומר זמן הריצה השתפר
גורם ההאצה הינו :

0.18667[sec/P) / 0.14933[sec/P) = 1.25

תרגול 6 (אמדל)

- שינינו מעבד כלשהו באופן הבא:
  - ההוראות שמטפלות בנקודה צפה (floating point) ירוצו פי 2.5 יותר מהר
  - פעולות גישה לזיכרון פי 3 יותר מהר
  - פעולות חיבור/חיסור בשלמים פי 1.5 יותר לאט
- מבדיקה עבור תוכנית מבחן עולה ש:
  - פעולות נקודה צפה תופסות 15% זמן מכלל תכנית המבחן
  - פעולות זיכרון 20%
  - פעולות חיבור/חיסור בשלמים – 40%
- כמה שיפרנו בסך הכל?

תרגול 6 (אמדל): תשובה

$$ExTime_{new} =$$

$$= ExTime_{old} \times \left[(1-(0.15+0.20+0.40))*1 + \left(0.15*\frac{1}{2.5}\right) + \left(0.20*\frac{1}{3}\right) + (0.40*1.5) \right] =$$

$$= ExTime_{old} \times \left[(1-(0.15+0.20+0.40)) + \frac{0.15}{2.5} + \frac{0.20}{3} + \frac{0.40}{(1/1.5)} \right] =$$

$$= ExTime_{old} \times 0.98$$

$$Speedup_{overall} = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{0.98} = 1.02$$

ניתן להכליל גם לחישוב CPI

$$CPI_{new} = CPI_{old} \times \left[(1 - Fraction) + \frac{Fraction}{Speedup(in\ cycles)} \right]$$

סיימנו...

שאלות?

אפקה
המכללה האקדמית
להנדסה בתל אביב