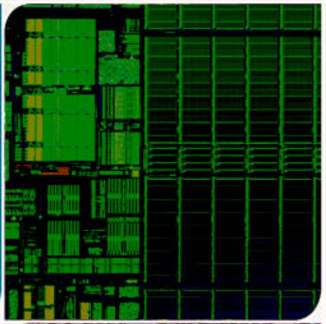
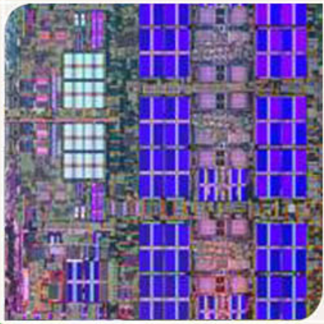
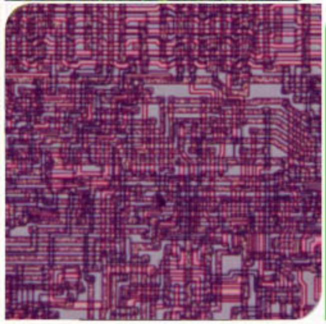
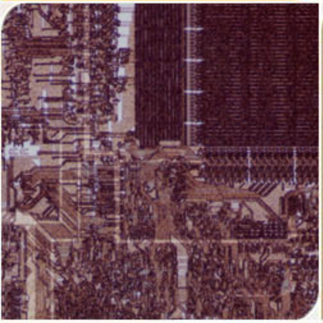






CS4290/CS6290

Fall 2011

Prof. Hyesoon Kim



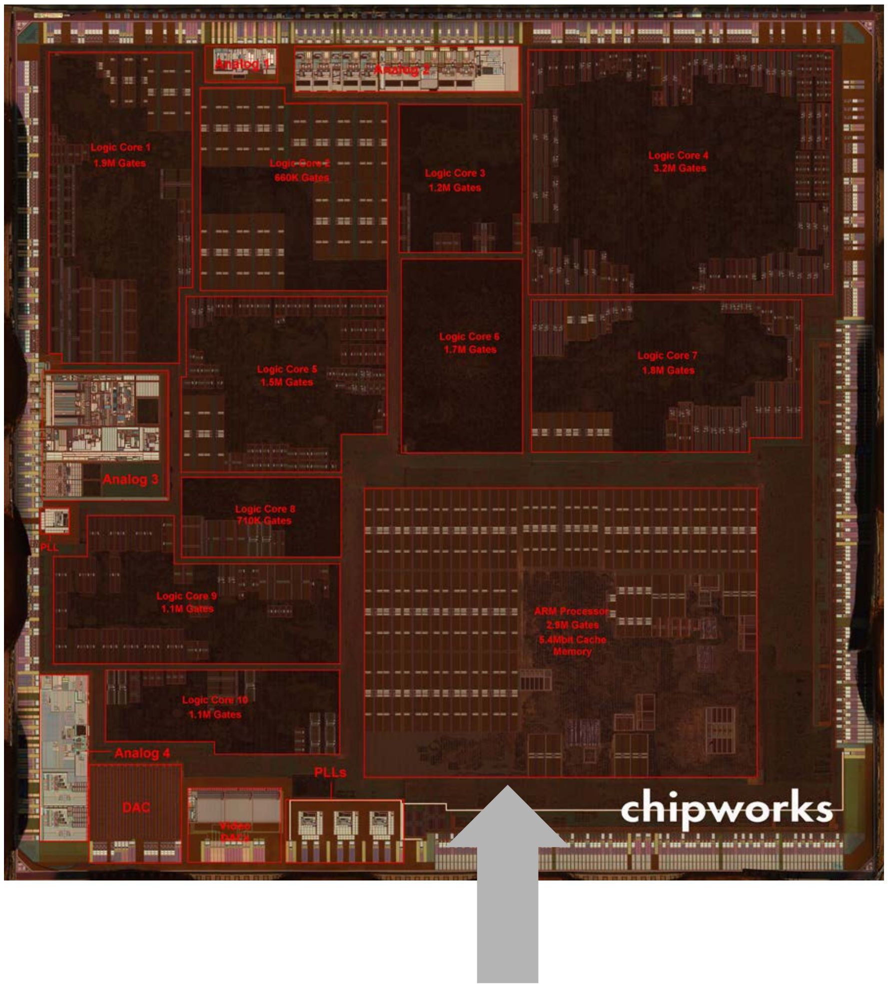
Georgia Tech College of Computing

Class Info

- Instructor: Hyesoon Kim (KACB 2344)
  - Email: hyesoon@cc.gatech.edu
- Homepage
  - http://www.cc.gatech.edu/~hyesoon/fall11
  - T-square (http://www.t-square.gatech.edu)
- Office hours: 3:00-4:30 Tu/Th
- TA: TBA
- Group mailing list: cs6290-2011@googlegroups.com
- Textbook: No required text book
  - Recommended book: Computer Architecture: AQA, **4th Edition** by Hennessy and Patterson
  - Jean-Loup Baer, *Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors, 1st edition*.
  - Papers

Floor Plan of A4 and A5→ iPhones/iPads

A4



A5

http://www.chipworks.com/en/technical-competitive-analysis/resources/technology-blog/2011/03/apple-a5-vs-a4-floorplan-comparison/

Georgia Tech | College of Computing

What is Architecture?

Problem
Algorithm
ISA
u-architecture
Circuits
Electrons

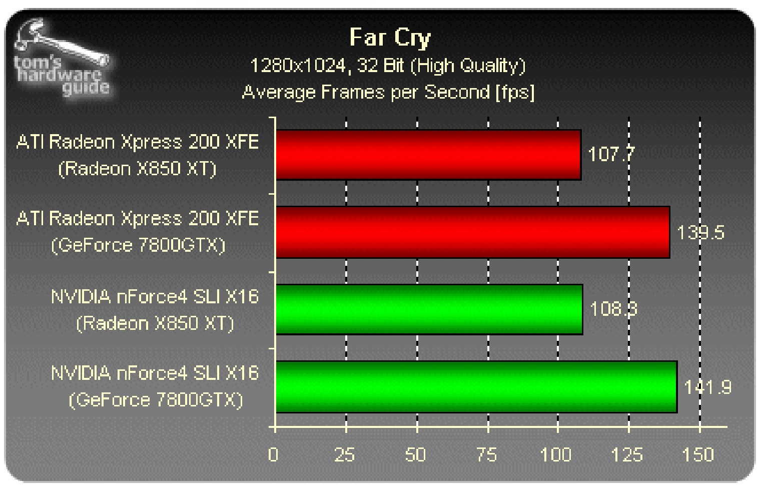
ISA: Interface between s/w & h/w

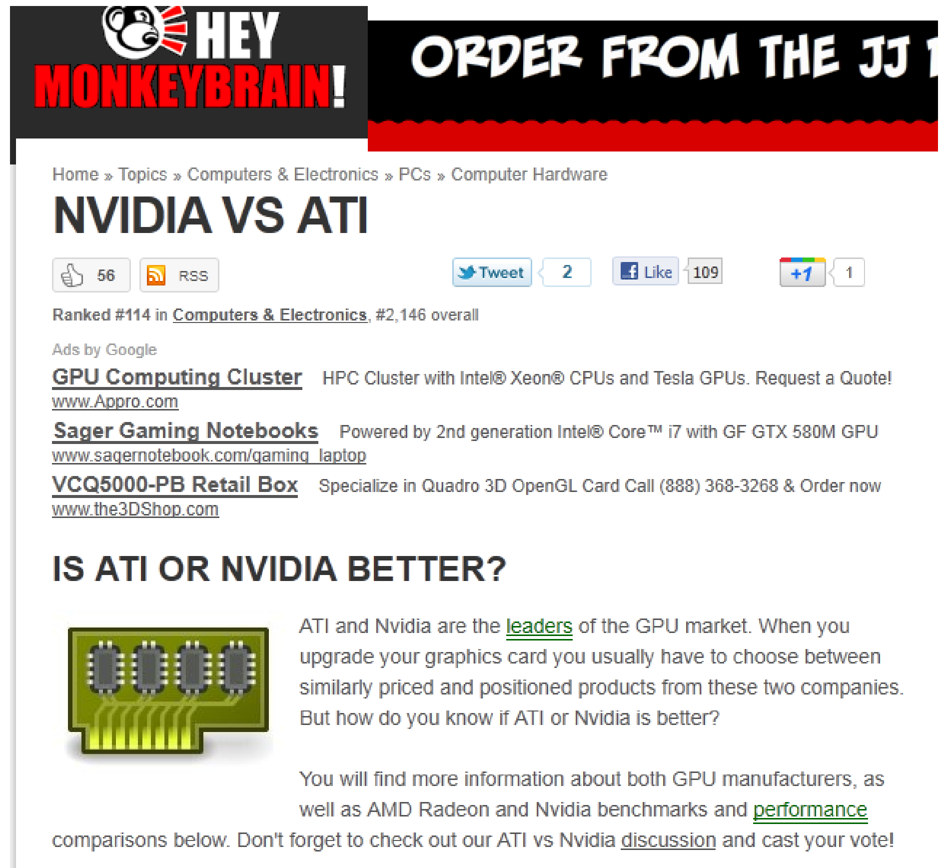
Warning!!

- This course requires heavy programming
- Don’t take too many program heavy courses!
- It is 3-credit course but you feel a 4-5 credit course
- The most ECElike course in CS

Architecture class

- can be fun or can be hard or look so easy…





HEY MONKEYBRAIN!

ORDER FROM THE JJ

Home » Topics » Computers & Electronics » PCs » Computer Hardware

NVIDIA VS ATI

56 RSS Tweet 2 Like 109 +1 1

Ranked #114 in Computers & Electronics, #2,146 overall

IS ATI OR NVIDIA BETTER?

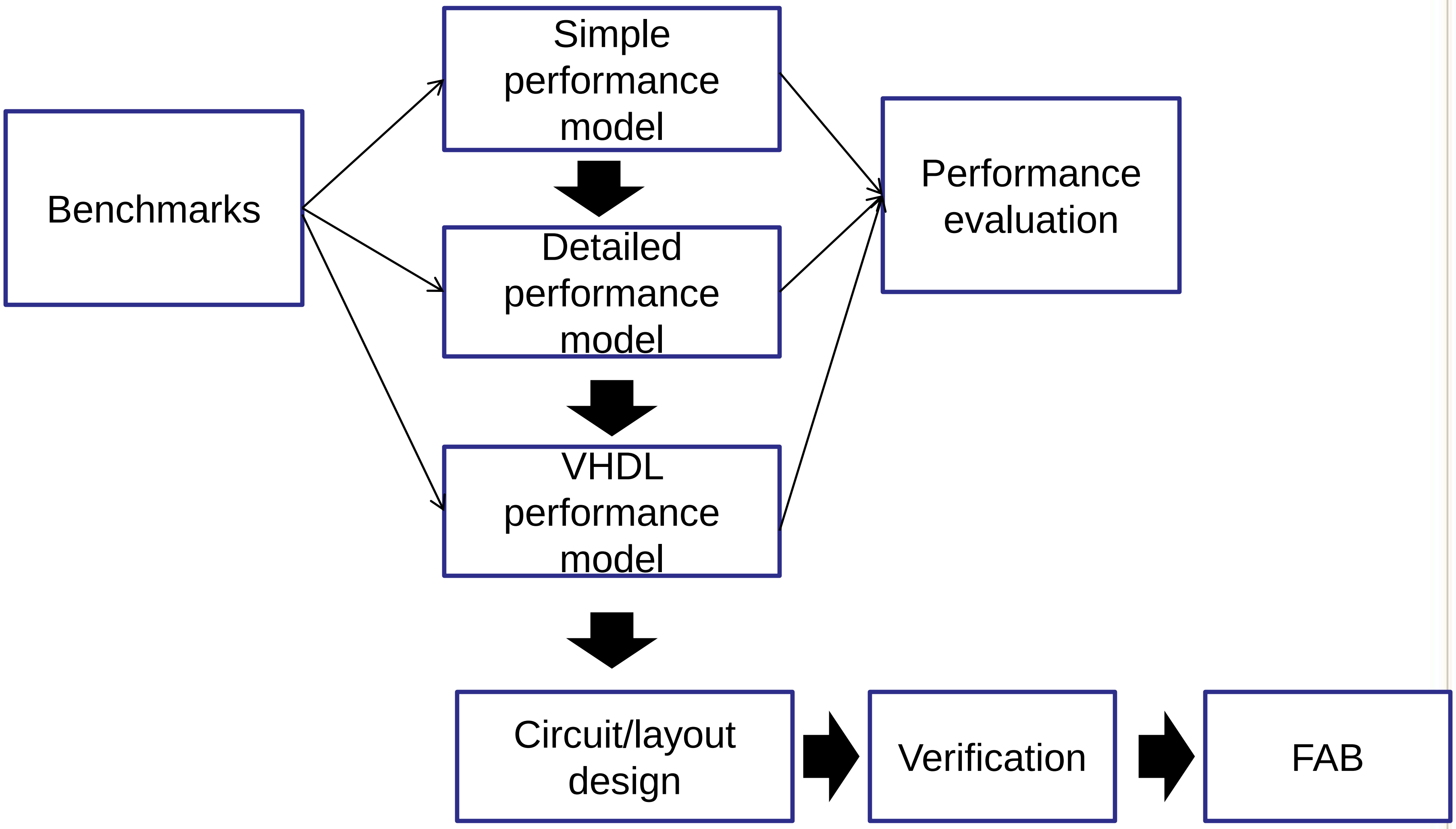
ATI and Nvidia are the leaders of the GPU market. When you upgrade your graphics card you usually have to choose between similarly priced and positioned products from these two companies. But how do you know if ATI or Nvidia is better?

You will find more information about both GPU manufacturers, as well as AMD Radeon and Nvidia benchmarks and performance comparisons below. Don't forget to check out our ATI vs Nvidia discussion and cast your vote!

Chip Design Process

- Select target platforms
  - Identify important applications
  - Identify design specifications (area, power budget etc.)
- Design space explorations
- Develop new mechanisms
- Evaluate ideas using
  - High-level simulations
  - Detailed-level simulations
- Design is mostly fixed→ hardware description languages
- VLSI
- Fabrications
- Testing

Architecture Study



Design Options

- Pipeline depth?
- # of cores?
- Cache sizes?, cache configurations? Memory configurations. Coherent, non-coherent?
- In-order/ out of order
- How many threads to support?
- Power requirements?
- Performance enhancement mechanisms
  - Instruction fetch: branch predictor, speculative execution
  - Data fetch : cache, prefetching
  - Execution : data forwarding

METRICS

Performance

- Two common measures
  - Latency (how long to do X)
    - Also called response time and execution time
  - Throughput (how often can it do X)
- Example of car assembly line
  - Takes 6 hours to make a car (latency is 6 hours per car)
  - A car leaves every 5 minutes (throughput is 12 cars per hour)
  - Overlap results in Throughput > 1/Latency

Measuring Performance

- Benchmarks
  - Real applications and application suites
    - E.g., SPEC CPU2000, SPEC2006, TPC-C, TPC-H, EEMBC, MediaBench, PARSEC, SYSmark
  - Kernels
    - “Representative” parts of real applications
    - Easier and quicker to set up and run
    - Often not really representative of the entire app
  - Toy programs, synthetic benchmarks, etc.
    - Not very useful for reporting
    - Sometimes used to test/stress specific functions/features

SPEC CPU (integer)

SPEC2006 benchmark description	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
	Benchmark name by SPEC generation				
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			

“Representative” applications keeps growing with time!

SPEC CPU (floating point)

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			

Spec Input Sets

- Test, train and ref
- Test: simple checkup
- Train: profile input, feedback compilation
- Ref: real measurement. Design to run long enough to use for real system
  - -> Simulation?
- Reduced input set
- Statistical simulation
- Sampling

TPC Benchmarks

- Measure transaction-processing throughput
- Benchmarks for different scenarios
  - TPC-C: warehouses and sales transactions
  - TPC-H: ad-hoc decision support
  - TPC-W: web-based business transactions
- Difficult to set up and run on a simulator
  - Requires full OS support, a working DBMS
  - Long simulations to get stable results

Multiprocessor's benchmarks

- SPLASH: Scientific computing kernels
  - Who used parallel computers?
- PARSEC: More desktop oriented benchmarks
- NPB: NASA parallel computing benchmarks
- GPGPU benchmark suites
  - Rodinia, Parboil, SHOC
- Not many

Performance Metrics

- GFLOPS, TFLOPS
- MIPS (Million instructions per second)

MIPS

Machine A with ISA "A": 10 MIPS

Machine B ISA "B": 5 MIPS

which one is faster?

Case 1

Alpha ISA

LEA A
LD R1 mem[A]
Add R1, R1 #1
ST mem[A] R1

→

X86 ISA

INC mem[A]

Case 2

Add, ADD, NOP ADD, ADD NOP, NOP ADD , NOP

CPU Performance Equation (1)

$$\text{CPU time} = \text{CPU Clock Cycles} \times \text{Clock cycle time}$$

$$\text{CPU time} = \text{Instruction Count} \times \text{Cycles Per Instruction} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}$$

ISA, Compiler Technology

Organization, ISA

Hardware Technology, Organization

A.K.A. The "iron law" of performance

CPU Performance Equation

$$\text{CPU time} = \text{CPU Clock Cycles} \times \text{Clock cycle time}$$

$$\text{CPU time} = \left(\sum_{i=1}^{n} IC_i \times CPI_i \right) \times \text{Clock cycle time}$$

For each kind of instruction

How many instructions of this kind are there in the program

How many cycles it takes to execute an instruction of this kind

CPU performance w/ different instructions

Instruction Type	Frequency	CPI
Integer	40%	1.0
Branch	20%	4.0
Load	20%	2.0
Store	10%	3.0

$$\text{CPU time} = \left(\sum_{i=1}^{n} IC_i \times CPI_i \right) \times \text{Clock cycle time}$$

Total Insts = 50B, Clock speed = 2 GHz

= (0.4*1.0 + 0.2*4.0+0.2*2.0 + 0.1*3.0) * 50 *10^9*1/(2*10^9)

Comparing Performance

- "X is n times faster than Y"

$$\frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$

- "Throughput of X is n times that of Y"

$$\frac{\text{Tasks per unit time}_X}{\text{Tasks per unit time}_Y} = n$$

If Only it Were That Simple

- "X is n times faster than Y *on A*"

$$\frac{\text{Execution time of app A on machine Y}}{\text{Execution time of app A on machine X}} = n$$

- But what about different applications (or even parts of the same application)
  - X is 10 times faster than Y on A, and 1.5 times on B, but Y is 2 times faster than X on C, and 3 times on D, and…

Which would you buy?

So does X have better performance than Y?

Summarizing Performance

- Arithmetic mean
  - Average execution time
  - Gives more weight to longer-running programs
- Weighted arithmetic mean
  - More important programs can be emphasized
  - But what do we use as weights?
  - Different weight will make different machines look better

Speedup

	Machine A	Machine B
Program 1	5 sec	4 sec
Program 2	3 sec	6 sec

What is the speedup of A compared to B on Program 1? 4/5

What is the speedup of A compared to B on Program 2? 6/3

What is the average speedup? (4/5+6/3)/2 = 0.8

What is the speedup of A compared to B on Sum(Program1, Program2) ?

(4+6)/(5+3) = 1. 25

Normalizing & the Geometric Mean

- Speedup of arithmetic means != arithmetic mean of speedup
- Use geometric mean: $\sqrt[n]{\prod_{i=1}^{n} \text{Normalized execution time on } i}$
- Neat property of the geometric mean: *Consistent whatever the reference machine*
- **Do not use the arithmetic mean for normalized execution times**

CPI/IPC

- Often when making comparisons in comp-arch studies:
  - Program (or set of) is the same for two CPUs
  - The clock speed is the same for two CPUs
- So we can just directly compare CPI's and often we use IPC's

Average CPI vs. "Average" IPC

- Average CPI = $(CPI_1 + CPI_2 + \ldots + CPI_n)/n$

- A.M. of IPC = ~~$(IPC_1 + IPC_2 + \ldots + IPC_n)/n$~~

Not Equal to A.M. of CPI!!!

- Must use *Harmonic Mean* to remain $\propto$ to runtime

IPC vs. Execution time

- A program is compiled with different compiler options. Can we use IPC to compare performance?
- A program is run with different cache size machine. Can we use IPC to compare performance?

Harmonic Mean

- H.M.$(x_1, x_2, x_3, \ldots, x_n)$ =

$$\frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3} + \ldots + \frac{1}{x_n}}$$

- What in the world is this?
  - Average of inverse relationships

A.M.(CPI) vs. H.M.(IPC)

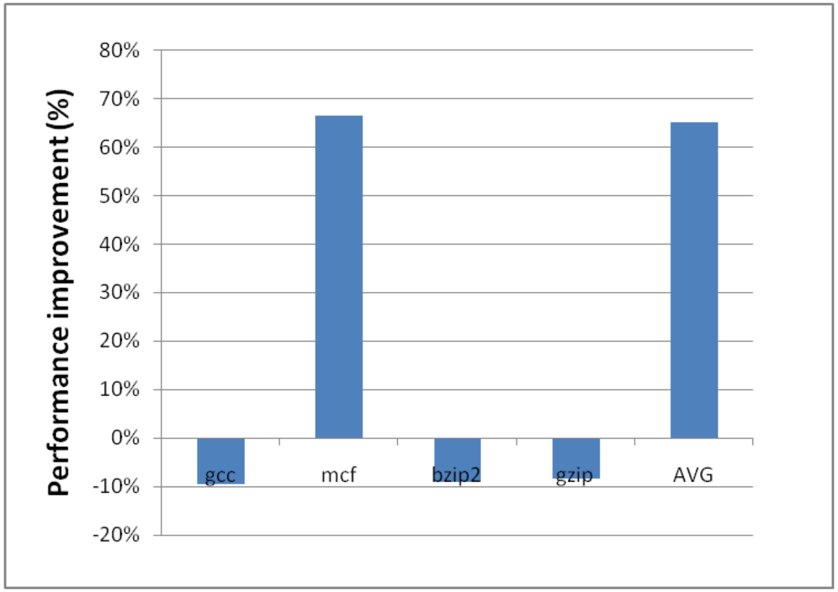
- "Average" IPC = $\dfrac{1}{\text{A.M.(CPI)}}$

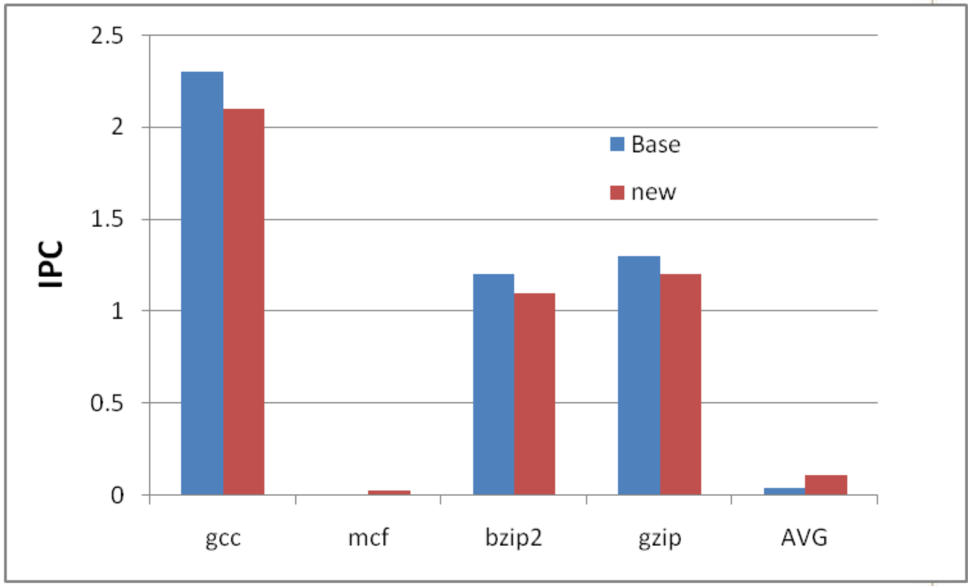
$$= \frac{1}{\frac{CPI_1}{n} + \frac{CPI_2}{n} + \frac{CPI_3}{n} + \ldots + \frac{CPI_n}{n}}$$

$$= \frac{n}{CPI_1 + CPI_2 + CPI_3 + \ldots + CPI_n}$$

$$= \frac{n}{\frac{1}{IPC_1} + \frac{1}{IPC_2} + \frac{1}{IPC_3} + \ldots + \frac{1}{IPC_n}} = \text{H.M.(IPC)}$$

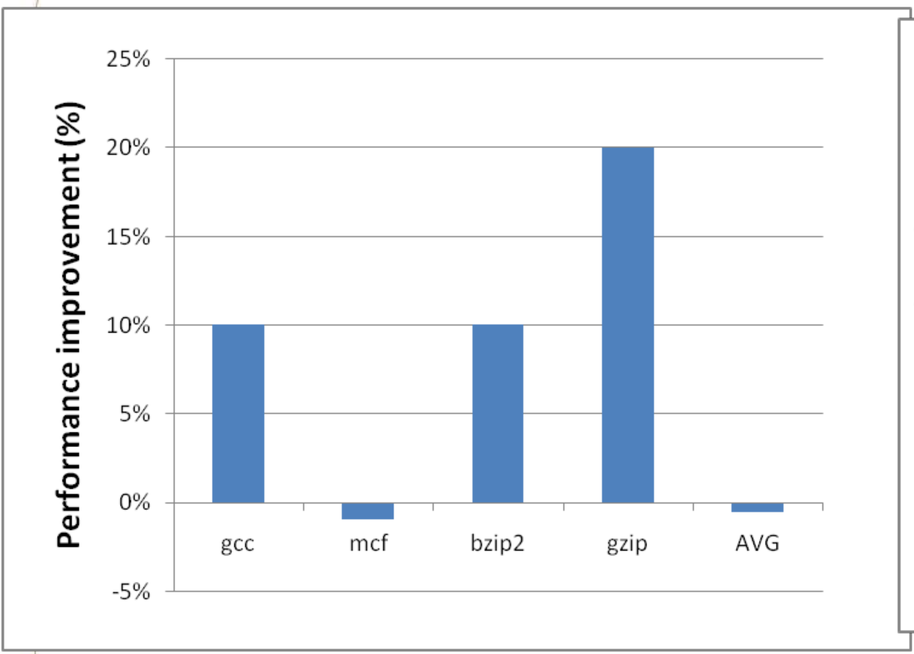
HMEAN's trick

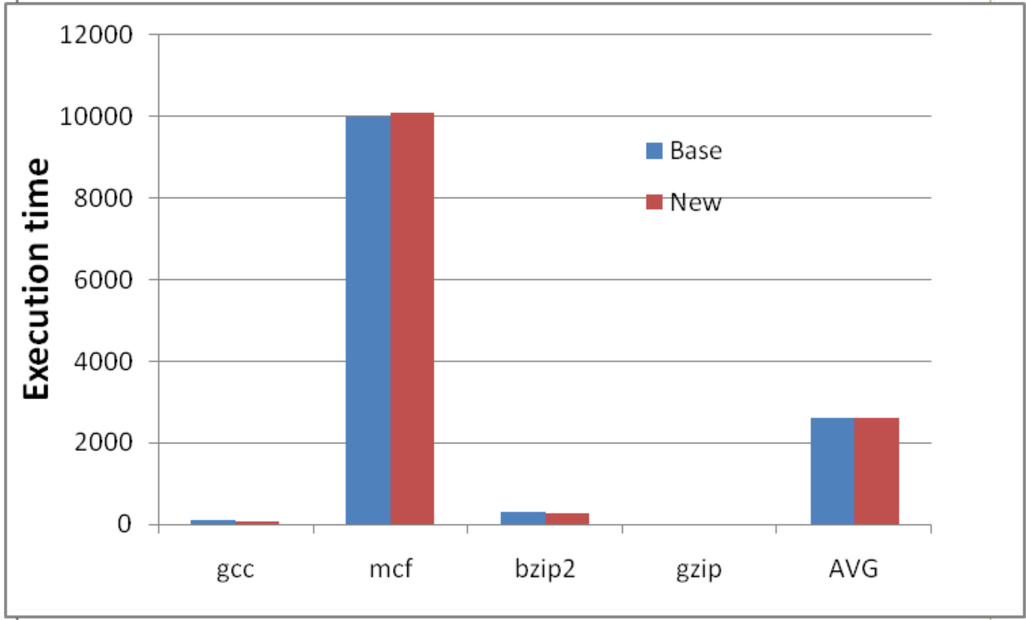




- One solution: use Gmean or show average without mcf and with mcf

AMEAN…

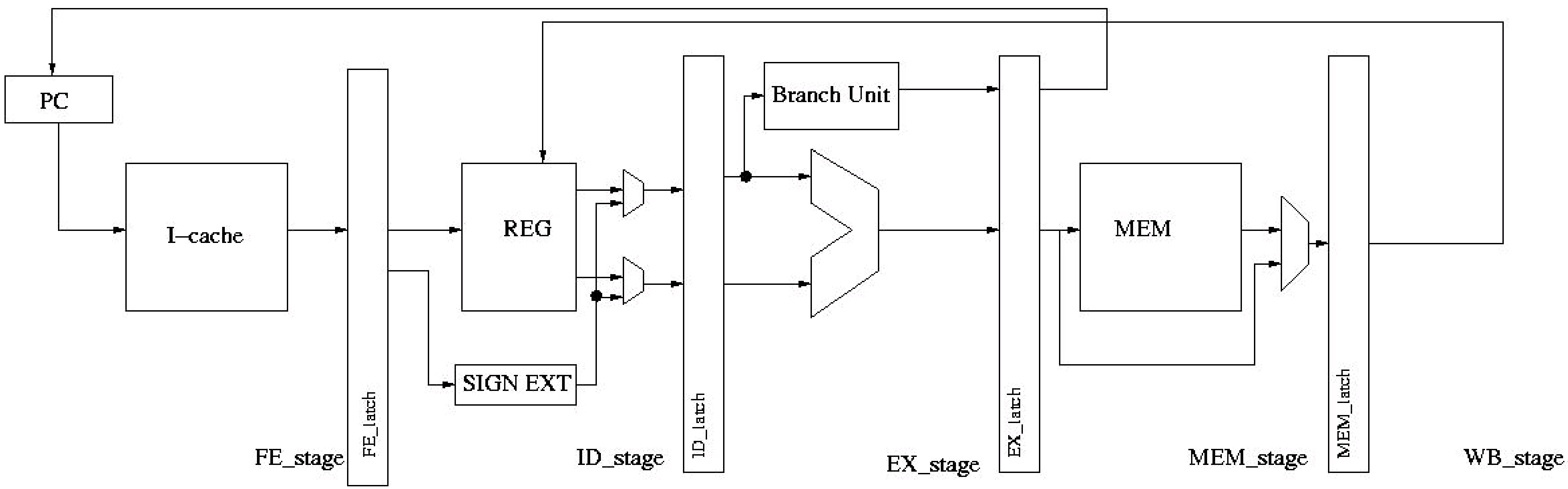




Sum(base)-Sum(new)/Sum(base) = -0.005%

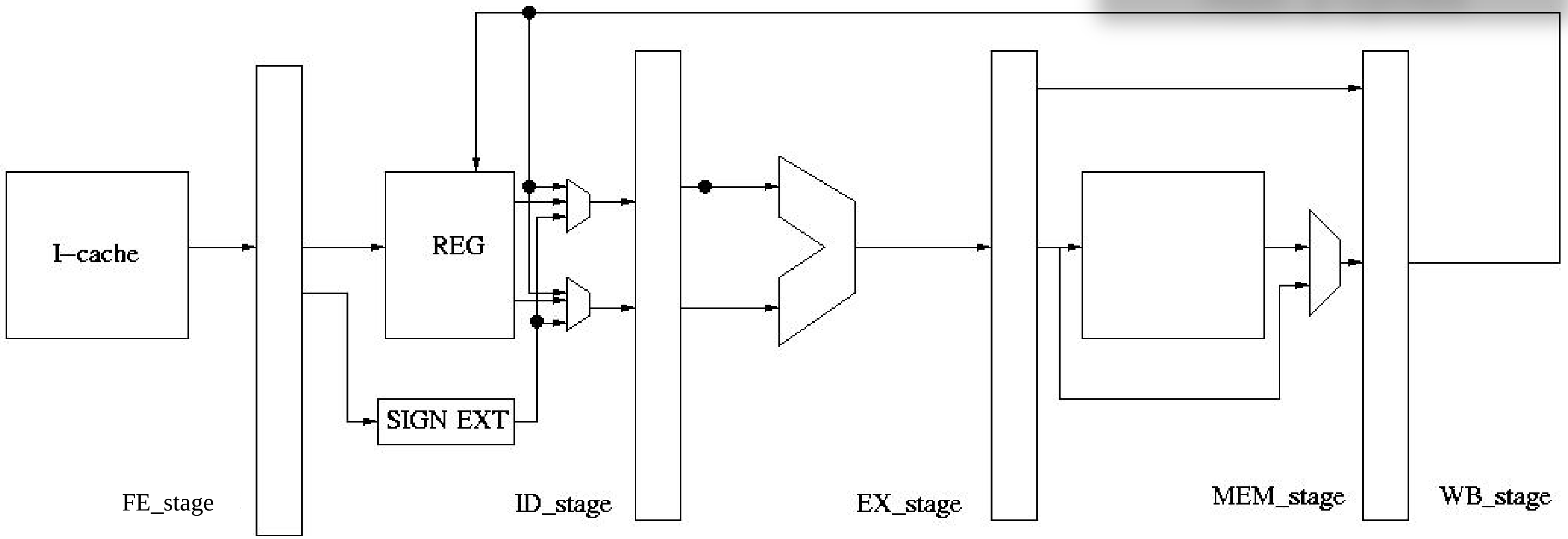
AVERAGE(delta) = 9.75%

Assignment #1



Dependent Instructions: dst data is available at WB

Add: 2 cycles



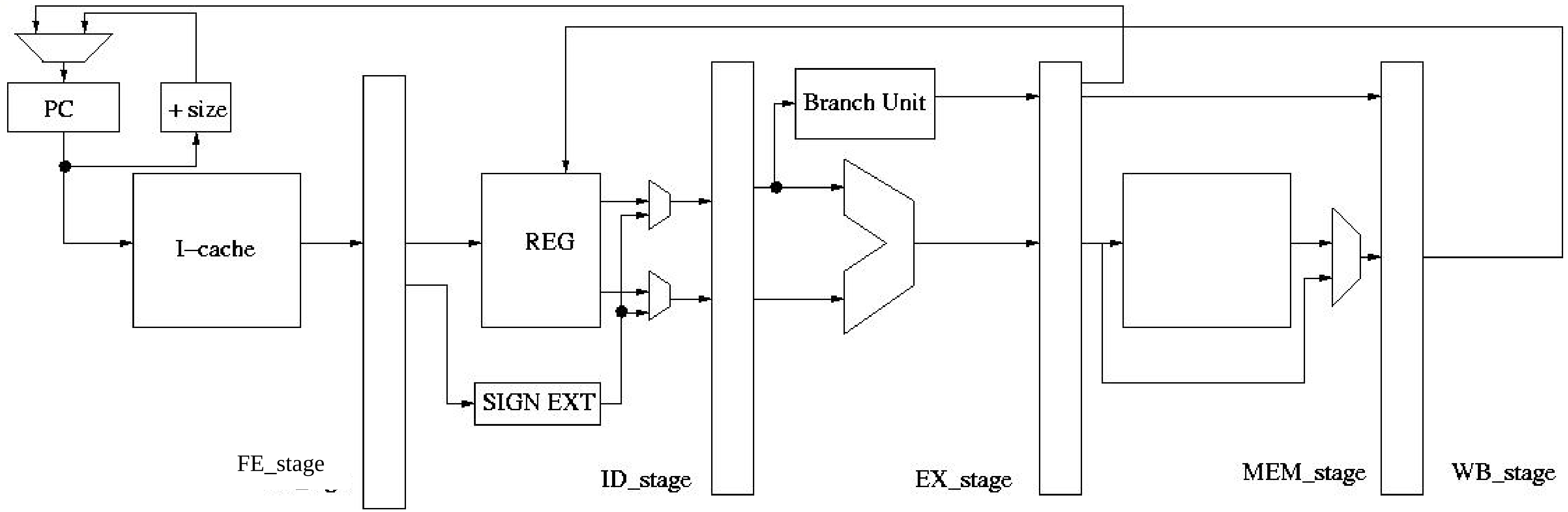

```
add  r1, r2, r3
sub  r4, r1, r3
mul  r5, r2, r3
```

FE	L	ID	L	EX	L	MEM	L	WB	L
add	add								
sub	sub	add	add						
mul	sub	sub	add	add					
mul	sub	sub		add	add				
mul	sub	sub				add	add		
mul	mul	sub	sub					add	

Handling Branches

```
       br     target     0x800
       add r1, r2,r3     0x804

target sub r2,r3,r4      0x900
```



cycle	PC (latch)	FE	ID	EX	MEM	WB
1	0x800	br				
2	0x804	add	br			
3	0x804	add		br		
4	0x804	add			br	
5	0x900	sub				br
6	0x904	add	sub			

Multicycle stages

Example: MIPS R4000