Putting it all together: G4 vs. K7

- CPU designers have to solve “conflicting” problems
  - Run programs as fast as possible
  - Maintain compatibility with previous versions of hardware
  - Use as few transistors as possible
    - Lower cost (=> more markets for processor?)
    - Lower power consumption
- Designers of Motorola G4 and AMD K7 took different approaches to balancing these tradeoffs
  - G4 focused on simplicity, lower cost
  - K7 focused on performance
- Examining tradeoffs can show how computer architects make decisions about how to build a CPU...

G4 vs. K7: basic differences

- Motorola G4 is relatively small and simple
  - 10 million transistors
  - 4 stage pipeline
  - Few addressing modes
  - Fixed length instructions => hardware instruction decoding
- AMD K7 is beefy and more complex
  - 22.5 million transistors
  - 12+ stage pipeline
  - Many addressing modes
  - Variable length instructions => “MacroOps” to decode them

G4 vs. K7: basic similarities

- Both use out-of-order execution
  - Reorder instructions for maximum performance
  - K7 has to work harder because of deeper pipeline
- Both use branch prediction
  - Accuracy more important for K7 because of pipeline depth
  - K7 devotes more space & effort to it
- Both use large caches
  - Caches reduce memory access time
  - Caches allow high-bandwidth access to memory
- Both have superscalar issue
- Both have vector units (though they handle vectors differently)

G4 instruction handling

- Read fixed length instruction from memory
- Decode using hardware
- Issue instruction to functional unit
  - Integer
  - Vector
  - Floating point
  - Branch unit
- Retire instructions in order

K7 instruction handling

- X86 instructions packed into predecode cache
  - Breaks byte stream into individual instructions
  - Deals with variable length instructions
- Transform x86 instructions into *MacroOps*
  - Done in hardware for simple instructions, microcode for complex ones
  - A MacroOp is composed of a register-register operation and/or a memory access instruction (called *ops*)
  - Decoding process takes 3 pipeline stages!
- Ops fed into execution pipeline
- Instructions retired in order

G4 vs. K7: functional units

- G4 functional units include
  - 1 FP unit
  - 2 vector units
  - 2 integer units
  - 1 address calculator
- Loads & stores
  - 6 entry scheduler
  - Limited out-of-order loads
  - 1 load or store per cycle
- FP execution speed limited
- Good vector execution speed

- K7 functional units include
  - 3 FP/vector units
  - 3 integer units
  - 3 address calculators
- Loads & stores
  - 44 entry scheduler
  - Flexible out-of-order loads
  - 2 loads per cycle
- Good FP execution speed
- Vectors somewhat slow

G4 vs. K7: math units

- G4 integer units (2)
  - One FU can do any operation
  - One FU can do only simple stuff
- G4 FP unit
  - Does all FP operations (no specialization)
  - Only allows one FP operation at a time

- K7 integer units (3)
  - Can do any operation
  - Up to 3 simultaneous operations
- K7 FP units
  - Does all FP operations (no specialization)
  - Also perform vector operations...

G4 vs. K7: vector units

- G4 vector units
  - 2 separate vector units
  - FP & vectors done by independent units
  - Vector unit operates on distinct register set
- G4 vectors are 128 bits long
- G4 faster at vectors because
  - Separate FU for vectors
  - Longer 128 bit vectors

- K7 vector units
  - Same units as FP units
  - Can't do both FP & vectors at the same time
  - Must reuse FP registers for vectors
- K7 vectors are 64 bits long
- K7 slower at vectors because
  - Shared FU with FP operations
  - Smaller vectors

G4 vs. K7: branch prediction

- G4 has relatively simple branch prediction
  - Less space required
  - May predict wrong more often
  - Penalty isn't so bad => pipeline is only four stages deep!
- Branch prediction takes the approach of improving penalties at the cost of reducing accuracy
- K7 has relatively complex branch prediction
  - Uses lots of space & transistors
  - Reduced misprediction rate
  - Penalty is high with a 10+ stage pipeline!
- Branch prediction takes the approach of improving accuracy at the cost of increasing penalties

Comparing design K7 and G4 approaches

- K7 takes the "complexity wins" approach
  - Throws transistors at the instruction decoding problem
  - Throws transistors into integer & FP functional units
  - Uses a superpipelined architecture: pipeline has relatively many stages, each of which is short
    - Clock speed can be faster
    - Hazards (data, control) cost much more
- G4 takes the "simplicity wins" approach
  - Keeps decoding simple
  - Relatively few integer & FP units, but higher utilization
  - Short pipeline resistant to hazards, but lower clock speeds
  - Considerably lower cost => broader markets

G4 vs. K7: which is better?

- So what's the bottom line?
  - Neither G4 or K7 is clearly better!
  - Each has its advantages and disadvantages
- K7 may be better for
  - FP intensive code
  - Code with relatively few (or predictable) branches
  - Systems where power & cost are less important
- G4 may be better for
  - Vector intensive code
  - Code with lots of branches and data hazards
  - Systems where power & cost matter more