Two main driving forces behind computer performance improvement : Improvements in semiconductor technology, causing systems to have faster clock speeds and cost less, Improvements in computer architectures, enabled by high level language compilers. Also led to RISC architectures. **Energy** IS a better metric to measure computer efficiency. **Higher** the clock frequency is, the more heat a computer generates. **Data-level parallelism (DLP):** arises because there are many data items that can be operated on at the same time. **Task-level parallelism (TLP):** arises because tasks of work are created that can operate independently and largely in parallel. **Instruction-level parallelism** exploits data-level parallelism at modest levels with compiler help using ideas like pipelining and at medium levels using ideas like speculative execution. **Thread-level parallelism**exploits either data-level parallelism or task-level parallelism in a tightly coupled hardware model that allows for interaction between parallel threads. **Request-level parallelism** exploits parallelism among largely decoupled tasks specified by the programmer or the operating system. **Power-Wall :** Limit on how much power a computer can consume, such that we have the ability to mitigate the heat generated in order to keep the computer running at safe temperatures. **Memory-Wall :** speed of main memory being improved slowly and taxes CPU speed because of memory access times. Slow speed of MM is limit on how fast computer can run. **Flynns-Taxonomy :** Single instruction stream, single data stream (SISD),Exploit ILP and TLP in some degree. Single instruction stream, multiple data streams (SIMD), Targets DLP, Graphics processor units. Multiple instruction streams, single data stream (MISD), No commercial implementation. Multiple instruction streams, multiple data streams (MIMD), Targets TLP and RLP. **Old view of comp arch :** focused on Instruction Set Architecture (ISA) design, and decisions regarding registers,memory addressing,addressing modes,instruction operands available operations, control flow instructions, instruction encoding. **New view of comp arch :** ISA design is less of a focus, Meet specific requirements of the target machine, Design to find a best trade off among performance, cost, power, and availability, etc, optimized for target applications, Consider ISA, micro architecture, logic/circuit design, implementation, etc. **DRAM :** dynamic random-access memory, this technology is the foundation of main memory. **Semiconductor Flash :** (electrically erasable programmable read-only memory) this nonvolatile semiconductor memory is the standard storage device in PMDs, and its rapidly increasing popularity has fueled its rapid growth rate in capacity. 8/10 times cheaper than DRAM, but order of magnatude slower. **Magnetic disk technology :** used for hard disk storage, 15/25 times cheaper than flash, 300/500 times cheaper than DRAM. **Transistors/Wires :** Transistor density grows exponentially, Moores law has been slowing down, Transistor performance scales linearly, Wire delay does not improve with feature size, its getting worse, on-chip interconnect design an important. **Bandwidth/Throughput :** Total work done in a given time,important for servers. **Latency/response time :** Time between start and completion of an event, important for individual users. Improvement in Bandwidth = square of improvement in latency. **Thermal Design Power (TDP) :** Characterizes sustained power consumption, Used as target for power supply and cooling system, Lower than peak power, higher than average power consumption. **Dynamic Energy :** $E = \frac{1}{2} CapacitiveLoad * Voltage^2$. **Dynamic Power :** $P = \frac{1}{2} CapacitiveLoad * Voltage^2 * FrequencySwitched$. **Relation of Energy and Power :** $Power = Energy * Frequency$. **Techniques for reducing power :** Turn off components not being used, Dynamic Voltage-Frequency Scaling, Low power state for DRAM, disks, Overclocking, turning off all but one core. **Clock frequency can be reduced dynamically to limit power consumption Energy efficiency has become a critical quality metric :** Measured by tasks/joule or performance/watt. **Static Power :** Transistors are not perfect switches, leakage current even when no switching. $Current_Static * Voltage$, scales w/number of transistors. **Reducing clock frequency reduces power, not energy. Factors Affecting Cost :** design one time cost, manufacturing and material recurring cost. **Cost driven down by learning curve, Yield: ratio between good products among all. Increase in volume reduces cost. IC wafer is large disk that contains multiple IC die. IC Dependability Modules :** Mean time to failure (MTTF) = 1/Failure Rate, Mean time to repair (MTTR), Mean time between failures (MTBF) = MTTF + MTTR, Availability = MTTF / MTBF. **To improve reliability: redundancy. Measuring Performance :** Response (execution) time and throughput. **Speedup of X relative to Y :** $\frac{exec\,time\,Y}{exec\,time\,X}$ **Wall clock time :** includes all system overheads, including memory and i/o. **CPU time :** Only the time the program spends in the actual CPU doing operations. **Principals of Computer Design :** Take Advantage of Parallelism, multiple processors, disks, memory banks, pipelining,multiple functional units. Principle of Locality,A property of programs: data and instructions typical reused. **Amdahls Law :** $ExecutionTime_{new} = ExecutionTime_{old} * (1 - Fraction_{enhanced} + \frac{Fraction_{enhanced}}{Speedup_{enhanced}})$ and $Speedup_{Overall} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$ **CPU time =** $instruction\,count * cycles\,per\,instruction * clock\,cycle\,time$ **Cycles per instruction =** $\frac{CPU\,clock\,cycles\,for\,program}{instruction\,count}$ **Dennard Scaling :** power density constant, as you increase number of transistors, they could go faster but use less power,voltage cant keep dropping and still maintain dependability of IC. **Amdahls Law :** practical limits to number of useful cores per chip, if 10 percent of task is serial then max performance benefit from parallelism is 10 no matter how many cores on chip. **ISA :** instruction set architecture is actual programmer visible instruction set of computer. **Comp Org :** includes high lvl aspects of computers design, memory system, memory interconnect, and CPU, now known as micro-architecture. **Improving Energy efficiency :** turn off clock of inactive modules to save power, slow the clock frequency down when there is no need to operate fast, low power mode for ram, use sensors to detect heat and slow down the machine. **Point of Failure :** every system is as strong as its weakest link, no matter how reliable one component is, whole system could fail due to something else. **Hardware enhancements that increase performance DO NOT increase energy efficiency. Cost of IC :** learning curve is change in yield, twice the yield will result in half the cost of IC. Cost of die reduces with how many dies per wafer and die yield. Dies per wafer measured with size of die and size of wafer. **Bandwidth has outpaced latency, band with grows by** $latency^2$**. feature size is min size of transistor or wire in x/y dimension. Two ways cpu's handle overheating :** circuity lowers clock rate, reducing power, if not enough then chip will power down. **Static Power :** $Power_{static} = Current_{static} * Voltage$. **Dark Silicon :** too many transistors that you cannot possibly turn them all off at the same time. **Principle of locality :** programs tend to reuse data and instructions that have been recently used. **Temporal Locality :** recently accessed items likely to be re accessed. **Spatial Locality :** items whose addresses are near one another tend to be referenced close together in time. **Amdahls Law :** performance improvement of faster mode is limited by the fraction of time the faster mode can be used. **1ns = 1GHz. CPU Clock Cycles =** $\sum_{i=1}^{n} InstructionCount_i * CPI_i$ then multiply by clock cycle time to get CPU time. **Mem Hierarchy :** when word is not found in

cache, it is fetched from lower lvl and placed in cache before continuing, multiple words in a block are fetched for efficiency reasons, each block includes tag to indicate mem address, motivation for using this is to improve AMAT and reduce energy consumption by exploiting locality. **Set Associative :** when block is brought to set associative cache it is placed in set which is result of block address MOD num sets in cache. **Write-Through :** updates item in cache and writes through to update main mem. **Write-Back :** only updates copy inside the cache, when the block needs to be replaced it is copied back to memory. **Types of Misses :** compulsory - very first access to a block at start up or when cache is flushed, capacity - need to remove block to replace it with another because cache is full, conflict - if not fully associative each set can only contain so many blocks and they must be discarded to make room. **Avg Mem Access :** Hit Time + (Miss Rate * Miss Penalty) **6 Basic Cache Opts :** larger block size to reduce miss rate but may increase miss penalty, bigger caches to reduce miss rate but may increase hit time, higher associativity to reduce miss rate but may increase hit time, multi-lvl cache uses small L1 to reduce HT and large L2 to reduce MR and MP, giving priority to read misses over writes to reduce miss penalty done with write buffer, avoid address translation during indexing of cache to reduce hit time. **AMAT of Multi-lvl Cache :** $HitTime_{L1} + MissRate_{L1} * (HitTime_{L2} + MissRate_{L2} * MissPenalty_{L2})$ **Virtual Index, physical tag :** uses page offset of virtual addr (low order bits) to access L1 while the addr translation is performed in parallel, aims to reduce HT of L1 cache. **Access time and cycle time :** Access Time - time between when a read is requested and a word arrives, Cycle Time - minimum time between unrelated requests to memory. **Difference in DRAM and SRAM :** with dram when data is read it must be written back, sram does not need to be written back to and thus access time is close to cycle time. **10 Advanced Opts :** Reduce Hit Time - use small and simple 1st lvl caches and way prediction, Increase cache bandwidth - pipelined cache/multibanked cache/non blocking caches, Reduce Miss Penalty - critical word first and mergeing write buffers, Reduce Miss Rate - compiler optimization's, Reduce miss penalty or miss rate via parallelism - hardware prefetching and compiler prefetching. **Higher Associativity :** causes a higher hit time, larger miss penalty rewards higher associativity. **Way Prediction :** extra bits kept in cache to predict the way or block within the set of the next cache accesses, multiplexer is set early to select a desired block, only used in associative caches. **Critical Word First and Early Restart :** CWF - request the missed word first from memory and send to processors when it arrives, let processor continue while retrieving rest of words in the block. ER - fetch words in normal order but as soon as requested word arrives send to processor and let it continue exec. Benifit of these are low unless block size is very large. **Write Buffer :** sits between processor and memory so that CPU can simply do its writes to the write buffer and continue and the WB will handle actually writing that data to memory. **Write Merging :** if cpu writes to WB and it contains another entry to the same memory location and if so the new data is combined with previous entry to reduce number of writes. **Compilier Prefetching :** The compiler can prefetch instructions to request data before the processors needs it, comes in 2 forms, register and cache prefetch. **TLB :** translation lookaside buffer, it is used to cache references recently used page table entries to reduce time to translate, it holds tag that contains portion of address and data portion holds physical page address. **ISA :** instruction set architecture, software between code and hardware, instructions processed in fetch, decode, execute, write back cycle. **What Makes Good ISA:** programmability,performance,implementability,compatability. **Execution Time =** $IC * CPI * cycleTime$. **RISC vs CISC :** CISC is complex instruction set computing, contains big heavyweight instructions (lots of work per instruction), low instruction count, higher cycles per instruction and seconds per cycle. RISC is reduced instruction set computer uses small simple instructions, low cycles per instruction and seconds per cycle, higher instructions per program. **Address Alignment :** specifies whether there are any boundaries for word addressing, it is aligned if Addr MOD size = 0. **Little vs Big Endian :** LE-least significant byte within a word is stored in smallest address. BE-most significant byte in smallest address. **Associativity and Tag Bits :** The higher the associativity, the more tag bits in memory addr. **Why HT of associative longer than DM cache :**In AC, tag comparison must be done b4 the data can be found in a block using offset, these two steps can be done in parallel in a DM cache, also tag comparison in AC takes longer. **Virtual Memory :** VM allows us to run more programs on system that we have physical mem to support, provides protection to processes. **Virtual Memory Translation :** In virtual memory, a virtual address from CPU is divided into virtual page number and page offset. Next, the virtual page number is used to index the page table to find the physical page number, which is then combined with the page offset for the physical address. **Why more Registers may be bad:** slower access time, more bits required in instruction, more saving and restoring. **Num of Address modes in ISA:** more bits required in the instruction encoding to encode all addressing modes.

Ex: DM cache has 64 blocks, 16bytes/block, what does address 1200 map to?(1200/16 mod 64). **Ex :** comparing cache performance with AMAT = $HT * cycleTime + MR * MP$ and CPU Time = $IC * ((CPI\ if\ given\ or, hitTime * cycleTime) + \frac{req}{inst} * missRate * missPenalty(ns))$. **Ex:** L1 Global miss rate is same as L1 Local miss rate. L2 Local miss rate is L2MR/L1MR, Global L2 miss rate is localMRL1*localMRL2. AMAT of multilvl cache uses local miss rate of L2. Global MR is independant of cache Lvl. **Multi-LVL AMAT Ex :** HTL1 = 1, MPL2 = 200, HTL2 = 10, L1 Global/Local MR = 0.04, L2 Global MR = 0.02. Calculate AMAT? First find Local MR of L2 which is LocalL2/LocalL1 = 0.02/0.04 = 0.5. Now calculate AMAT = $1 + 0.04 * (10 + 0.5 * 200 = 5.4$. **DM Cache Ex :** what cache block will block 10 from main mem go if DM cache has 8 blocks?(10 MOD 8 = 2, so block 2). How about if its 2-way set associative w/8 blocks?(10 MOD 8 = 2, so block 2). **Tag,Index,offset Ex :** suppose 32 bit wide address, each cache block is 16 bytes, and the cache has 64 blocks, show ranges for parts of address. (index a block = 2^6 = 64 so 6 bits(4,9), offset = 2^4 = 16 so 4 bits (0,3), left over 22 bits for tag. **Effective CPI of L1 w and w/o L2 cache :** ECPIWO = idealCpi + $\frac{req}{inst}$ * MRL1 * MPL1. ECPIW = idealCpi + $\frac{req}{inst}$*MRL1*(HTL2+MRL2*MPL2). **Always use local miss rates** **Cache Block Size in bytes:** L1 data/8 or $2^{block\ offset}$. **Size of L1 and how to increase it:** L1=$2^{page\ offset}$ **Page Size in bytes:** PS=$2^{page\ offset}$.