Computer Architecture Final Exam

Metrics

Performance metrics:
Clock cycle time CCT = 1/Clock Frequency
IC: Instruction Count; CPI: Cycle per instruction
Execution time = Num Cycles × CCT = IC × CPI × CCT

- Latency, Throughput:
- Inter-task parallel only improves throughput;
- Intra-task parallel improves both

Power and Energy

- Dynamic power: $CV_{dd}^2 f$
- Static power: $V_{dd} I_{\text{leakage}}$

Power limited by infrastructure, its density limited by thermal dissipation, energy limited by battery capacity.
Scalability: Amdahl's law
Speedup = $((1-f) + f \times \frac{1}{S})^{-1} = (1 - f + \frac{f}{S})^{-1}$
Means: amean for absolutes; hmean for rates; gmean for ratios

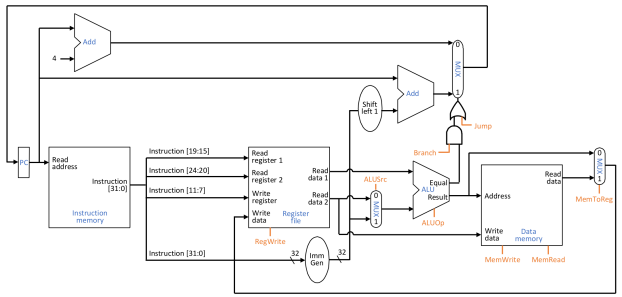
ISA

Calling Convention
Control: jump and link; return: jr ra
Register convention: caller saved: ra, t, a;
callee saved: sp, s.
Switch case with jump tables: compute and use jump register.
Short constants directly use immediates, large constants takes more steps to create.
Encoding: R,I,S,B,J,U

- Designed mostly consistent with each other, reduce hardware overhead; quickly extract which registers to read
- Branch: small distance (13); jump for long distance

Parallelism
SIMD: Data-level, increase processing throughput, amortize instruction cost.
MIMD: Thread-level. Shared-memory multi-core processors.
Data race: solved by atomic instructions.

Single Cycle Processor



CPI = 1, but long critical path, hardware underutilized.
Pipeline: $f = (\frac{t_{comb}}{k} + t_{setup})^{-1}$, also latency $t_{comb} + k \times t_{setup}$ getting worse.
Balanced: Pipeline throughput is limited by the slowest stage.
Miminize register width: where to partition the combinational logic.

Pipeline Processor and advanced

IF, ID, EX, MEM, WB. Reasons:

- Roughly balanced in latency
- Natural functionality, with small amount of signals across stages
- Not too many stages with too high register latency overheads

Control: Propagate control signals until consumed.

CPI = base CPI + stalls per instruction.
Stalls are caused by structural, data or control hazard.
Structural: Instructions always use each resource once and in the same stage, e.g. half cycle Read/write; writeback at 5-stage.
Data: Only Read after write hazard.
Forwarding: reduce all R-format, 1-stall for load-use.
Compiler optimizations: filling load delay slots with independent instructions.
Control: Put branch, jump to ID stage.
Branch prediction at IF stage. (BHT saturate counter+ BTB)
Exceptions and Interrupts
goto OS handler. Save PC and exception/interrupt reasons.
Precise Exception: All previous instructions completed, no offending instruction and the following instructions were started.
ILP = T_1/T_∞. ILP is scope dependent: Larger instruction scheduling windows or Out-of-order execution.
Renaming solves all WAW, WAR dependencies.
WAR: 2 completes not before 1 dispatches.
WAW: 2 completes not before 1 completes.
Fetch, Decode, Dispatch/Rename, Issue, Execute, Commit

Cache Hierarchy & Main memory

Cache Banking: each bank is like an individual cache
block address determines which bank a block should be stored:
Tag + Index + Bank ID + offset.
A block can be in any way of a set, but only go to one bank.
Shared data in private caches: **cache coherence** issues. Use protocols such as single-writer multiple-read(SWMR).
3-state protocol Modified/Shared/Invalid. M: one cache has valid and latest copy, and can write. S: caches and memory have valid copy.
Cache optimization: Non-blocking

- Allow for hits while serving a miss (hit-under-miss)
- Allow for hits to pending misses (hit-to-miss)
- Allow more than one miss (miss-under-miss)

Cache optimization: Data Prefetching
Predict and prefetch: reduce cold misses
Extra stream buffers to hold the prefetched data
Use Miss Status Handling Register (MSHR) keep track of miss
Other optimization: Loop reordering, loop blocking (split matrix to small pieces)

Cache

Memory hierarchy: appear to be large and fast
Virtual memory: appear to be contiguous and private
Locality: programs work on a small portion of data at any time. Temporal vs. Spatial
Memory hierarchy: holds more-often-access data on smaller and nearer devices
Cache:
Data transferred in unit of **cachelines**.
Cache hit: *Latency = time to access cache*, so average memory access time: *AMAT = hit latency + miss rate × miss penalty*
Miss reasons: compulsory/cold, capacity, conflict
Tradeoff of block size: increasing miss penalty convex miss rate.
Exploit spatial locality when size is small, sacrifice temporal locality when size is large with small number of blocks.
Direct mapped:
Index: the only choice for each location - middle $n - m$ bits.
Tag: identify which the exact location - upper $32 - n$ bits.
The rest m bits for each 2^m bytes block.
Cache entry = data block + tag + metadata.
Overhead = tags + metadata = $(1 + 32 - n) \times 2^{n-m}$
Total storage = $(2^m * 8 + 1 + 32 - n) * 2^{n-m}$.
N-way set associative cache: to reduce conflict misses
Index $n - m - w$ bits, tag $32 - n + w$ bits. Replacement.
Write policies: write through vs. write back
write back: maintain a dirty bit, write back to memory when dirty data is replaced. Use write buffer to avoid stalls for write-through.

DRAM & Main memory

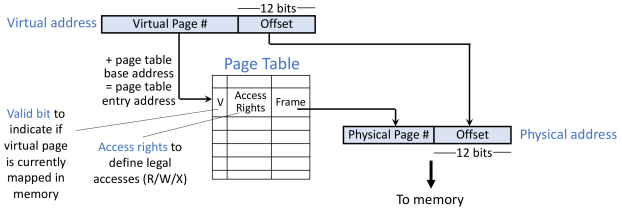
DRAM Bank: A cell array: $2^n \times 2^m$ 2D array
IO width: w bits. Row buffer: $2^m \times w$ bits. Each access:

- Activate: transfer an entire row to row buffer
- Read/Write: Read or write columns in row buffer
- Precharge: Clean up row buffer for next Activate

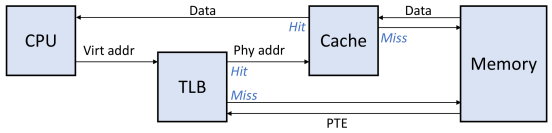
Three cases: row miss, row hit and row conflict.
Closed-page vs. Open-page policy (Precharge or not).
Bank-level parallelism: multiple banks on one chip. Spread access to different banks, overlap latencies
Dual Inline Memory Module: multiple narrow-IO chips form a wide interface. Same latency; higher capacity and bandwidth, but higher power; fixed granularity.
Channel-level parallelism: multiple independent channels. Non-uniform memory access.

Virtual Memory

By memory hierarchy with caches, fix memory latency problem.
A large contiguous, private memory illusion → Virtual memory.
Virtual Memory: a level of indirection through a page table
Physical memory shared by virtual memories.
Page table: a page table base register
One page table entry (PTE) per virtual page number



Synonym: a process with different virtual addresses point to a same physical address. Copy: only change frame number and mark the page as write-protected.
Homonym: different processes can use the same virtual address but point to different physical addresses.
Page sharing: different processes can share a page by setting virtual addresses to point to the same physical address.
Table size: Virtual addresses N bits, pages 4KB. Then total number of pages: 2^{N-12}, total page table size: $2^{N-12}\times$PTE size. (May be to big for memory). This is proportional to virtual address size but not physical address size.
Solution: **hierarchical page table**, only top level must be stored in memory. (A sparse tree that save page table size).
TLB: (translation look-aside buffer) caches for page table.
Issue 1 Context Switching: add a process ID (PID) in each TLB entry, allows entries from multiple processes to co-exist
Issue 2 Limited TLB Reach: solved by multi-level TLBs, larger pages or multiple page sizes.
If cache physically tagged and indexed, not parallel. If virtually indexed, translation & cache access in parallel.



Disks

Hard Disk Organization Disk → platters $\xrightarrow{2}$ surfaces → tracks → sectors
Access time: Control + queuing + seek time (move head to track)+ rotational latency + data transfer time
Small data: dominated by seek time and rotational latency
Data locality and OS scheduling: better performance since large sequential access amortizes seek and rotational overheads
Solid State Disks (SSD or flash):

- Data are read/written in units of pages.
- To write new data, must erase first (in units of blocks).
- A block wears out after certain number of writes.
- No in-place modification in SSD/Flash.
- FTL: maintain logical to physical page mapping.

I/O

Software interfaces: **register-based**, **memory-mapped I/O**
Hardware designs: **buses**, **switched interconnects**

Software Interface

Register Interface
Data registers: TxD, RxD
Command/status registers: Rd, Wr (control), TxRdy, RxRdy (status), ECC(errors)
Memory-Mapped IO:
Physical address space assigned to each I/O device. This I/O address space is protected by the address translation mechanism. (Must ask OS to map the I/O addresses to its virtual memory space)

Hardware designs

Bus
Master-slave arrangement. For multiple masters, **arbitration** is used: one device granted access at any time.
Typical buses: processor-memory, processor-processor; I/O buses connect devices to the processor-memory bus.
Synchronous vs. Asynchronous
Data skew: clock different between two parallel clock signals
Clock cycle time must $\gg$ skew time, limits bus frequency.
Serial bus use a single wire but with no data skew. (higher frequency but needs (de)serialize data and commands)
Pipelined bus pipeline request and response.
Split-transaction bus allows out of order response.
IO Notification: Polling or interrupt

- Polling: check a status register periodically
- Works great if I/O rate fixed or needs frequent attention
- Interrupt works best when I/O rate is unpredictable

Direct Memory Access
Achieves large data transfer without bothering the processor, DMA works as an I/O device itself.
Issue 1: OS may swap pages out to disk. Solution: memory pinning
Issue 2: contiguous virtual address may not be physically contiguous!
Solution 1: chain series of single-page requests; single interrupt at the end
Solution 2: DMA engine uses virtual addresses
Issue 3: data involved in DMA may reside in processor cache
Hardware routes memory accesses for I/O through cache

GPU

GPU Thread Model
Single program, multiple data (SPMD)
Single instruction, multiple threads (SIMT)

- Large off-chip memory bandwidth
  - And maintain sufficient pending memory accesses
- Tolerate long latency by switching between warps
  - Similar to multithreading

GPU provides **Offloading Computation**, introducing data transfer overhead.
GPU characteristics:
Larger register files, Smaller and shallow cache hierarchy.
Throughput-oriented, Latency is less a concern than bandwidth
Main memory: limited capacity, but extremely high bandwidth

Custom Hardware

Tradeoff: Flexibility vs. Efficiency
Memory access: Maximize reuse using buffer.
Line buffer for image processing;
Double buffer: two banks switch roles between fetching data from memory and serving data to computation.

Summary

1. **Parallelism:** multi-core processors, pipeline processor, channels/banks of DRAM
2. **Pipeline:** Higher Throughput but increased latency
3. **Out-of-order Execution:** dynamically discover parallelism.
4. **Speculation:** beyond dependency, guess the output. Must also include: checking and recovering. E.g., Branch prediction, Data prefetch.
5. **Locality → Caching:** Instruction/data cache, translations (TLB)
6. **Indirection:** Virtual address, Flash Translation (FTL), Renaming OoO
7. **Amortization:** Cacheline access, domain-specific designs.
8. **Common case fast:** optimize the dominant portion, but do not over optimize (Amdahl's law)