Powers of 2:

- $1, 2, 4, 8, 16, 32(5), 64, 128, 256, 512, 1024(10)$
- Prefixes (by powers of $2^1 0$): kibi(10), mebi(20), gibi(30), tebi(40), pebi(50), exbi(60), zebi(70), yobi(80)
- Split power of two into tens and ones, use first line to get number, second line to get name. ie, $2^{23} = 2^3$ mebi = 8 mb.

MIPS Basics:

- 32 registers, 32 bits for data or instructions
- R instruction: 0 opcode, 2^6 funct codes. 61 regular, 3 for srl, sll, sra
- I instruction: 61 opcodes, mirror of R. branch limits: PC - $2^1 7$ to PC + $2^1 7$ -4.
- J instruction: 2 opcodes (j, jal), target is PC[32:29](target ¡¡ 2). limits: $2^2 8$ (?)
- Declarations. Text = machine code , Data = binary rep of data , word = 32 bit quantities in order.
- Memory - code at bottom, then static, then heap (grows up). Stack at top, grows down.
- RTL - reg trans lang - "madd rd, rs, rt" –¿ R[rd] = Mem[R[rs]] + R[rt]; PC = PC + 4

Gotchas:

- Remember to sll by 2 when adding counters to pointers - we need word alignment!
- Iterative vs recursive - good compiler can optimize well, so
- Prologue and Epilogue - save ra and all non temp vars to stack, then restore.
- BNE has range 16+2, since last 2 bits are zeros. Also, 2's complement.

Bit Twidling:

- AND : 0, 0, 0, 1. used to turn bits OFF (X AND 0)
- OR : 0, 1, 1, 1. Used to turn bits ON (X AND 1)

Caches:

- Compute cache size = $2^(indexbits + rowbits)$
- Temporal locality - youngest element likely touched again
- Spatial locality - memory reads are often sequential or at least close in space.
- Dirty bit for write-back, valid bit for tag.
- Direct Mapped - each memory location associated with exactly ONE location - ie, 4 spots and mod 4.
- N-Way Set Associative - map to rows, but rows have N blocks. 2-way gives great performance boost, avoid ping pong. Remove oldest elem. (LRU)
- Fully Associative - blocks go anywhere. Hard to find stuff.
- Offset - # bytes is ln width of cache (4 words, 2 bits)
- Index - # bytes is ln # rows in cache (32 rows, 5 bits)
- Tag - what remains. (32 - 2 -5 = 25 bits.)
- Write through - write to cache and memory
- write-back - write to memory on flush, uses dirty bit.
- bits = tag + valid bit + (dirty bit) + data (8*B)

Miss Types:

- Compulsory - cache empty, so everything misses (cold start)
- Conflict - two blocks map to same space, so forced to swap out
- Capacity - cache is small, so we often run out of space.

Running a Program (CALL):

- Compile - High level to low level. C → MIPS.
- Assemble - Outputs Object code and information tables. Replaces pseudo instructions.
- Link - Input: object file header, text, data, reloc info, symbol table, debug info. Output executable code.
- Load - Allocates space, copies code and static vars into mem, init regs and set stack pointer.

Map reduce

- Map - Takes in data, emits intermediate (key, value)
- Combiner - combines values from map into lists, so less spamming of reducer
- Reduce - takes (key, value[]) and simplifies down into (key, result). Writes to output.

Datapath

- 5 steps in MIPS cpu - IF = instruction fetch, ID = instruction decode, EX = execution(Mem-ref - calculate address, Arith-log : perform operation), Mem = load and store data from and to memory, WB: Write Data Back to Register
- 3 types of hazards
- 1. Structural Hazards - Prevent next logical instruction in next clock cycle. Hazards that occur due to competition for the same resource (register file read vs. write back, instruction fetch vs. data read). Caching and clever register timing can solve these hazards.
- 2. Data Hazard - Need to wait for previous instruction to complete its data read/write - alu results. Hazards that occur due to data dependencies (instruction requires result from earlier instruction). These are mostly solved by forwarding, but lw still requires a bubble.
- 3. Control Hazard - Deciding on control action depends on previous instruction - jumps, branches. Hazards that occur due to non-sequential instructions (jumps and branches). These cannot be solved completely by forwarding, so were forced to introduce a branch-delay slot (MIPS) or use branch prediction.
- Bubbles - Need to send in a nop, all 0's. Used to stall.
- Flush - happens when branch prediction fails, have to kill currently executing inst.
- Pipelining - split up actions into stages, each taking around the same time. Latency is sum of all steps, throughput is single step. Compare this to single cycle, where you can stretch on step and still be okay.

GPU

- Throughput optimized. Many slow cores. Errors happen but not critical. Lots of mem bandwidth. terminology
- Have to load data into GPU mem, using CUDA.
- Each function is called a kernel, called with a specific geometry. SIMD.
- Threads organized into 3D blocks. Then organized into 3D grid.
- Have to check bounds constantly, otherwise massive errors.

OpenMP

- Uses blocks with common commands - pragma omp parallel (command_here)
- Dataraces - without a critical block or manual intervention, can read and write at the same time.
- fork-join model - main process forks into parallel, then joins back up.
- Good on multicore machines.

Boolean Logic

- Laws: Or is +, AND is *. Commutative, distributive.
- $\neg AA = 1$, $\neg(X + Y + Z) = \neg X\neg Y\neg Z$, $X(X + Y) = X$, $Y(X + \neg Y = XY)$
- $\neg(A + B) = \neg A\neg B$
- Converting from state machine to formula - list state bits and input, then output and next state bits. Write formula for every output that's a one, then simplify with boolean laws.

combinational logic

- draw an OR, NOT, NOR, AND, etc. - know symbols
- How to get clock rate - find longest path between two registers, and convert to Hz (1/freq)

Flynn Taxonomy

- {Single, Multiple} Instruction {Single, Multiple} Data.
- SIMD and MIMD most common today.

Virtual Memory

- page size 4KiB
- pages all same sizes, saved to disk into swap
- TLB - usually 128 entries - set or fully ass.
- Offset - log page size.
- Page Table - holds all entries of VA's - map to either mem or disk. So size is based on virtual mem, not phys.
- Step by step process - get addr, ignore offset, look at index. Check TLB, then check Page Table. If in memory, page table says where, otherwise, page fault and go to disk. Use offset to pull out byte from page.
- Ways to improve rates - make TLB larger, make Phys mem larger, make memory/disk faster.
- VA is split into VPN and page offset. PA is also split into PPN and page offset. page offset is same for both, since page size is equal.
- log(page size) = page offset bits. The rest of the bits are for VPN and PPN.
- Page table holds VPN, valid bit, dirty bit, permissions, and PPN. This is a mapping. Index = VPN.
- TLB (Translation Lookaside Buffer) holds MRU VPN tags. Usually small, around 128 entries. Set or fully ass.
- Page table can have at most # phys pages valid entries.

IO

- polling - every n time units, check ready bit. formula: (poll/s * clocks/poll)/(total clocks)
- polling disk - (mb/s) / (b/poll) = (poll/s), then (poll/s)*(clocks/poll), then all over (total clocks / sec).
- interrupts - proceed as normal, then respond to an interrupt. formula: (% active) * (X Mb/s) / (X Bytes/interrupt) = (interrupts / second). take that, and multiply by (clocks / interrupt) , and then take that over (total clocks / second).
- Memory mapped IO - bits of memory mapped to a device. We write bits to 0xFFFF000, get data sent to device. dgaf how. (abstraction)

Random

- AMAT = Time for a hit + Miss rate * Miss Penalty
- Strong scaling is when adding more machines makes your algorithm faster on a given data.
- Weak scaling is when adding more machines lets you process more data in the same amount of time.
- Amdahls law = $\frac{1}{(1-P)+\frac{P}{S}}$
- malloc - n*sizeof(datatype), not just n.

1. Relax. This test will not matter in 5 years.
2. You survived 61a and 61b, you can do this too.
3. The "A" is yours.

CS61c Final Study Guide, by Ivan "Vania" Smirnov
BTC donation address: 14i5jja4KLoUD9jTetbz4AEtTV9Ud1WpsW