1 Ch 1 stuff

- [Speedup] $= \frac{\text{[Latency 1]}}{\text{[Latency 2]}} = \frac{\text{[Throughput 1]}}{\text{[Throughput 2]}}$
- [Performance Improvement] = [Speedup] − 1
- [Exec. Time] = [Instr. Count] × [CPI] × [Clock Speed]
- [Performance] = [Execution Time]$^{-1}$
- [Dynamic Power] $\propto$ [Activity] × [Capacitance] × [Voltage]^2 × [Frequency]

2 ASM

- Calle saves $s# registers, caller saves everything else
- Stack grows down

```
.data
prompt:          .asciiz "Ente...rs: \n"
input:           .space 81
inputSize:       .word 80
exitMessage_p1:  .asciiz "Word count: "
.text
```

3 Binary Stuff

Float

"But we're not dealing with real numbers, this is floating point baby!"

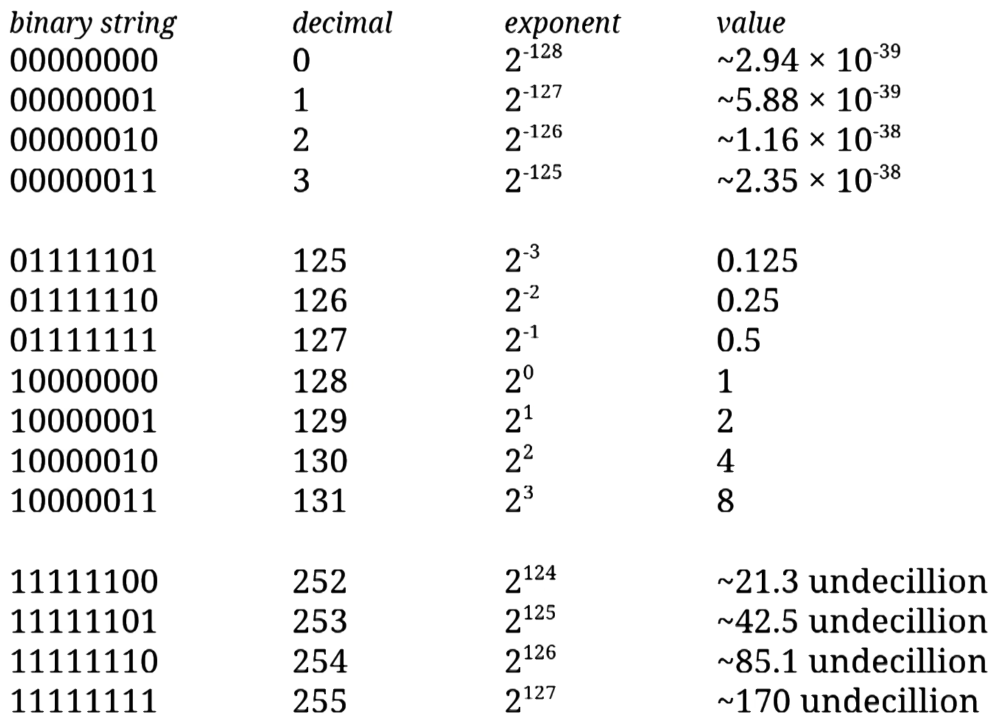
- Dec from Float: $(-1)^{\text{[Sign bit]}} \times (1 + \text{[Mantissa]}) \times 2^{(\text{[Exponent Bits] - [Bias]})}$
- Float from Dec: Convert to bin sci notation, 'sub

$$thebias$$

$$toget$$

  in' to floating exponent land
- Dec to F32:
- Zero: "00000000" exponent, all zero mantissa, sign as usual
- Inf: "11111111" exponent, all zero mantissa, sign as usual
- NANs: "11111111" exponent, sign and mantissa "left to implementer's discretion"
- Subnorms: "00000000" exponent, then replace the leading 1 with a zero and continue as usual. Getting increasingly smaller but less precise

Exponent chart from jan masali:



binary string	*decimal*	*exponent*	*value*
00000000	0	2^{-128}	~2.94 × 10^{-39}
00000001	1	2^{-127}	~5.88 × 10^{-39}
00000010	2	2^{-126}	~1.16 × 10^{-38}
00000011	3	2^{-125}	~2.35 × 10^{-38}
01111101	125	2^{-3}	0.125
01111110	126	2^{-2}	0.25
01111111	127	2^{-1}	0.5
10000000	128	2^{0}	1
10000001	129	2^{1}	2
10000010	130	2^{2}	4
10000011	131	2^{3}	8
11111100	252	2^{124}	~21.3 undecillion
11111101	253	2^{125}	~42.5 undecillion
11111110	254	2^{126}	~85.1 undecillion
11111111	255	2^{127}	~170 undecillion

Demorgans and Boolean algebra

- Xors are 1 if there's an even number of 1s, 0 if an even number. – Thus 'Parity'
- $\neg(A \vee B) = \neg(A) \wedge \neg(B)$
- $\neg(A \wedge B) = \neg(A) \vee \neg(B)$
- You can flip the operation and flip weather they're internally or externally negated.

4 FSM

Not that hard, just like, make sure all states and lines can be justified and are labeled.

5 Pipelining (oh god)

Usual steps of a 5 stage pipeline

- IF Instruction fetch 1 cycle, get next instruction from InstrMem or cache
- DR Data read from register
- AL ALU, does the computation
- DM Data memory:
- RW Read Write

Bypassing

- Point of production:

  add, sub, etc: end of ALU

  lw: end of DM – the mem it's reading into register

- Point of consumption:

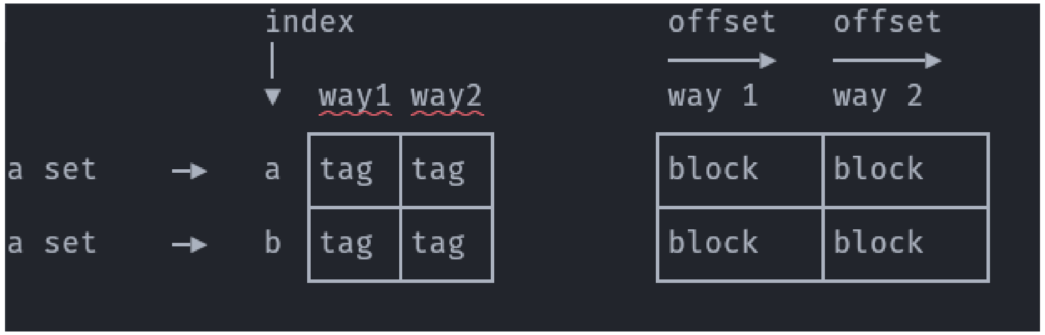
  add, sub, lw: start of ALU

  sw/lw \$1, 8(\$2):

  start of ALU for \$2

  start of DM for \$1

6 Cache



- Offset and Index need the bits they need, tag gets the rest

  Offset is for within the block

  Index: which set we're referencing

  - Tag array size = sets * ways * tagSize
  - [Offset] = address%[block size in bytes]
  - [Index] = (address/[block size in bytes])% $\log_2$(Sets Count)
  - [Tag] = address/([block size in bytes] $*$ $\log_2$([Sets Count]))
  - Tag bits: remaining bits
  - Offset bits: depending on blocksize, determines what byte within the block is being referenced $\log_2(blocksize(B))Indexbits : determinedbyamountofsetslog_2(setcount)$

- Usually we write-allocate, briging a miss into cache. Read misses always bring block into cache

  Usually we evict the least-recently used block

- Bit string: tag index offset

7 Spectre/Meltdown

- "Spectre refers to a whole family of potential weaknesses of which meltdown is just one" - Computerphile video description
- These are the result of a combination of speculative execution and out of order execution
- Meltdown is a specific kernel memory exploit
- Need a: "lw \$t1, 0(secret); lw \$t2, \$t1" series of two instructions to leave footprints in cache

8 Virtual Memory

- Memory virtualised by the kernel, the program thinks it has a single uninterrupted area to work with
- Programs have a pagetable, which is a list of how virtual pages are mapped to physical pages in memory
- Page table translations are mostly done via the Translation Lookaside Buffer, which falls back to the pagetable

9 Multiprocessor Design

- Each core has a cache, the LLC is shared
- TODO: Protocol descripions and examples of that chart

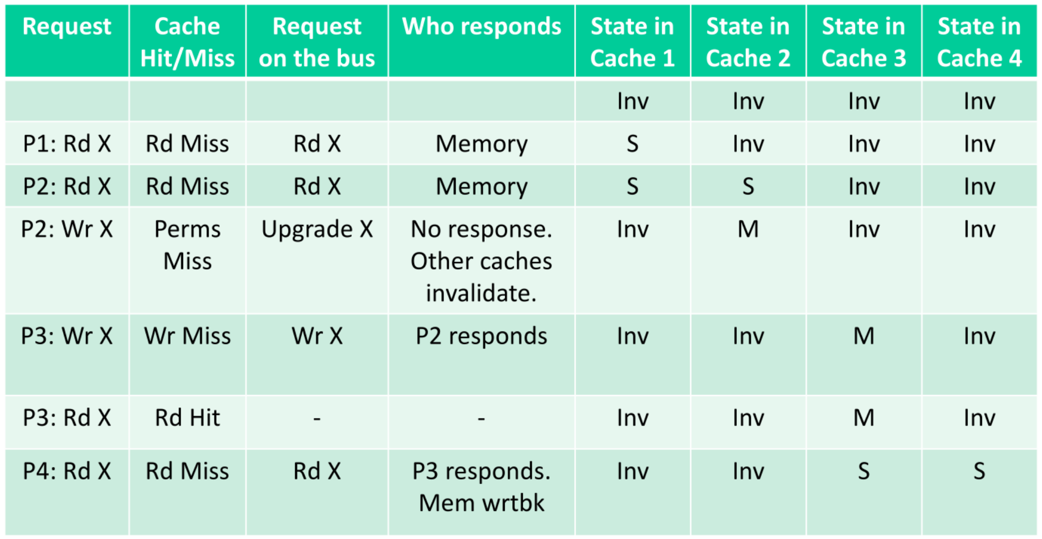
Cache Coherence Protocls

- Directory-based: A single location (directory) keeps track of the sharing status of a block of memory
- Snooping: Every cache block is accompanied by the sharing status of that block - all cache controllers monitor the shared bus so they can update the sharing status of the block, if necessary
- Write invalidate: a processor gains exclusive access of a block before writing by invalidating all other copies
- Write-update: when a process or writes, it updates other shared copies of that block

Locks

- Atomic exchange: simulatneous load and store operation, locks memory at the same moment it reads. – locks it while it transfers into register

MP exmaple



Request	Cache Hit/Miss	Request on the bus	Who responds	State in Cache 1	State in Cache 2	State in Cache 3	State in Cache 4
				Inv	Inv	Inv	Inv
P1: Rd X	Rd Miss	Rd X	Memory	S	Inv	Inv	Inv
P2: Rd X	Rd Miss	Rd X	Memory	S	S	Inv	Inv
P2: Wr X	Perms Miss	Upgrade X	No response. Other caches invalidate.	Inv	M	Inv	Inv
P3: Wr X	Wr Miss	Wr X	P2 responds	Inv	Inv	M	Inv
P3: Rd X	Rd Hit	-	-	Inv	Inv	M	Inv
P4: Rd X	Rd Miss	Rd X	P3 responds. Mem wrtbk	Inv	Inv	S	S

10 Basic structure of a GPU:

- many Single Instruction Multiple Thread cores, each with several warps and a warp scheduler. – large cache
- Very quick to abandon the current project if it stalls and start work on the next – minimal downtime, easy to context switch
- Each SIMT core has priavate L1 cache, large L2 shared by all cores. Each L2 bank services a subset of addresses
- The

11 Disk stuff:

- 1-12 platters (glass disks), 5-30k tracks (rings), 100-500 sectors, 512B/sector (circle around a track)
- Arm moving to correct track ⇒ seek time, 5-12ms, maybe less
- Rotational latency: time to rotate sector under head, usually 2ms
- Transfer time: time taken to transfer a block of bits out of disk, usually 3-65 MB/s
- Disk controller: maintains disk cache, sets up transfers.
- Mean time to Failure/Restore

  Reliability MTTF

  Availability: fraction of time service matches specs, MTTF / (MTTF + MTTR)

12 Raid types overview:

- Raid 0: No redundancy, stripes disks across drives to improve parallelism.
- Raid 1: Mirrors all disks, can sometimes increase read speed, highly redundant
- Raid 2: bit level striping, requires lockstep drives. Lots of parity drives
- Raid 3: byte level striping with dedicated parity disk. Highest sequential read speeds. Usually requires lockstep drives.
- Raid 4 and 5: block level striping, 4 has a dedicated parity disk, 5 distributes parity sections among drives.
- Raid 6: Same as raid 5, but has two parity blocks per stripe, again distributed among drives. Can work even after two drive failures. Technically raid 6 can have an arbitrary number of parity blocks added for increased redundancy.