Two's Complement

- Flip bits. add 1.

Integer Multiplication

$1000 = 8_{10}$
$\times 1001 = 9_{10}$

1000
+1000

$1001000 = 72_{10}$

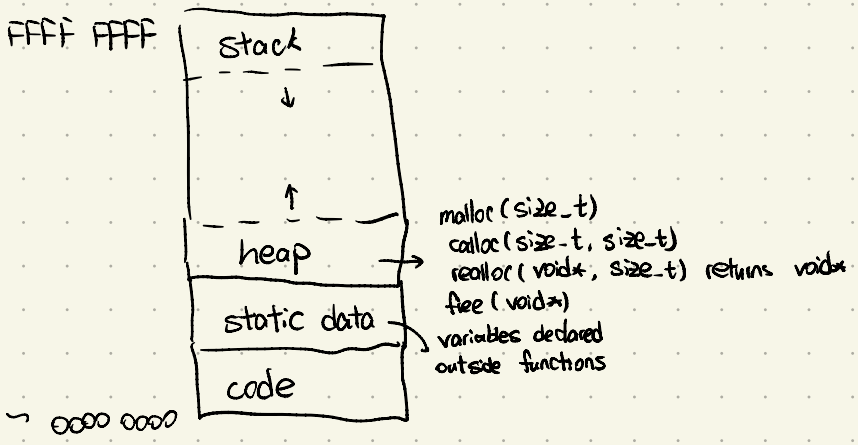
Division

Quotient: 1001
Dividend: 1001010
Divisor: 1000
−1000
1010
1000
10 → Remainder

Floating Point.

	S	E(xponent)	S(ignificand)
	1 bit sign	Biased Integer	Integer
32b	1 bit	8 bits (−127)	23 bits
64b	1 bit	11 bits (−1023)	52 bits

Memory



Boolean Algebra

$x\bar{x} = 0$	$x + \bar{x} = 1$
$x0 = 0$	$x + 1 = 1$
$x1 = x$	$x + 0 = x$
$xx = x$	$x + x = x$
$xY = Yx$	$x + Y = Y + x$
$(xY)z = x(Yz)$	$(x+Y)+z = x+(Y+z)$
$x(Y+z) = xY + xz$	$x + Yz = (x+Y)(x+z)$
$xY + x = x$	$(x+Y)x = xY$
$\bar{x}Y + x = x + Y$	$(\overline{x+Y}) = \bar{x}\bar{Y}$
$(\overline{xY}) = \bar{x} + \bar{Y}$	

S=0, E=0, S=0 ⇒ +0
S=1, E=0, S=0 ⇒ −0
S=?, E=1111…, S=0 ⇒ +/− ∞
S=?, E=111…, S=Not 0 ⇒ NaN.
S=?, E=0, S=Not 0 ⇒ Denorm

Denorm: bias +1
Implicit exponent $-(2^{e-1}-1)+1$ ⇒ −126 in 32b, −1022 in 64b
with implicit leading 1 changed to an implicit leading 0.

Transistor energy efficiency

$P = \frac{dE}{dt}$, $E_{0\text{-to-}1} = E_{1\text{-to-}0} = \frac{1}{2} C V_{dd}^2$

$P_{sw} = \frac{1}{2} \alpha C V_{dd}^2 F$ → clock frequency (Hz)
↑ total capacitance to be switched
↑ operating voltage
"activity factor" = average percentages of capacitors switching per cycle.

$E_{sw} \propto V_{dd}^2$, $\gamma_{logic} \propto V_{dd}$ ⇒ Improve by Vdd and compensate performance by adding cores.

Cache

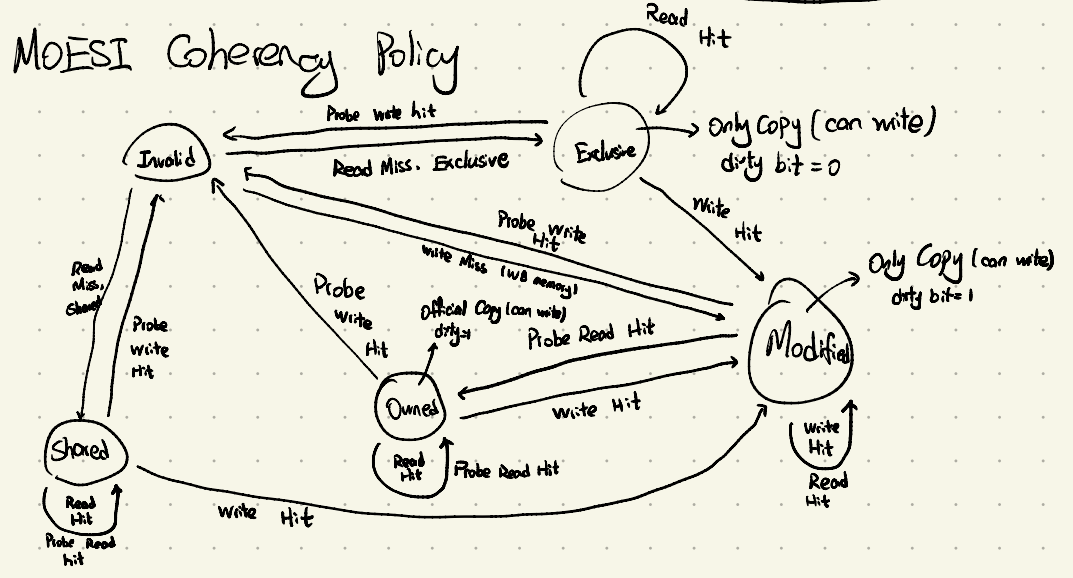
- Temporal 时间 Locality & Special 空间 Locality
- | Tag | set Index | Byte offset |

Extra
1. valid bit
2. Dirty bit
3. Shared bit

Misses
1. Compulsory (first time)
2. Capacity
3. Conflict (solved with fully associative)
4. Coherency

Performance:
AMAT = Hit Time + Miss Rate × Miss Penalty.

MOESI Coherency Policy



Cache Design Choices

1. Increasing Associativity
   - Hit time ↑↑ from DM to ≥ 2 ways (since mux correct way to processor)
   - Hit time ↗ (slightly increased) for further increase in associativity
   - Miss rate ↓ ∵ reduced conflict misses (1→2→4)
   - Miss penalty mostly unchanged since replacement policy runs in parallel with fetching missing line from memory
2. Increasing # of entries.
   - Hit time ↑ ∵ reading tags and data from larger mem structures
   - Miss rate ↓ ∵ reduced capacity and conflict miss
     Miss rate ↓ ∩ 2x / capacity ↑ ∩ 4x
   - Somepoint Increase in hit time may overcome improvement in hit rate.
3. Increasing block size.
   - Hit time unchanged but may be slightly reduced as # of tags is reduced
   - Miss rate ↓ at first (spatial locality), then ↑ because of conflict miss↑
   - Miss penalty ↑ but with fixed constant initial latency it is amortized over the whole block.
4. Add a "victim cache"

CSR: (trap handler ↑, place to store memory →)
sepc / scause / stvec / sscratch / satp

$$\text{Speedup} = \frac{\text{Time w/o E}}{\text{Time w/ E}}$$

$$\text{Time}_{w/E} = \text{Time}_{w/o\ E} \times \left[(1-F) + \frac{F}{S}\right]$$

$$\text{Speed up} = \frac{1}{(1-f) + \frac{F}{S}}$$

CALL (Compiles ⇒ Assembler ⇒ Linker ⇒ Loader)

Compiler ← Computationally Heavy

1. Lexer: input → tokens
2. Parser: tokens → abstract syntax tree
3. Semantic Analysis and Optimization
4. Code Generation.

Assembler

- foo.s → foo.o
- Reads and uses Directives
- Produces machine lang.
- takes two pass (remember → fill) to solve forward reference problem.
- Outputs Symbol Table and Relocation Table.

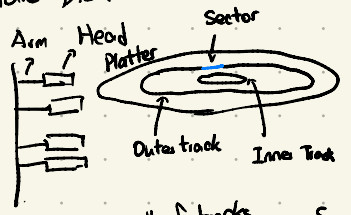
Linker.

Puts multiple .o file → .out file

Loader.

- Reads header.
- Creates new address space
- Copy instructions and data to mem
- Copies arguments onto stack
- Initiates registers (sp)
- Jumps to start-up routine
  - copies arguments from stack → reg
  - sets PC.
- Main routine exits ⟶ Cleans up
- Links DLL.

Hard Disk



$$\text{Seek Time} = \frac{\#\text{ of tracks}}{3} \cdot \frac{S}{\text{track}}$$

Rot Time = ½ time of a cycle.

MTTF / MTTR / MTBF = MTTF + MTTR

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF+MTTR}} = \frac{\text{MTTF}}{\text{MTBF}}$$

RAID 0: Stripping

RAID 1: Disk Mirroring

RAID 3: Single Parity Disk (bitwise)

RAID 4: high I/O Rate Parity (blockwise)

RAID 5: high I/O Rate Interleaved Parity

RAID 6: + parity block per stripe ↑

Warehouse

$$\text{PUE (Power Usage Effectiveness)} = \frac{\text{Power of total building}}{\text{Power of IT equipment}}$$