COMPUTER SYSTEMS

EQUATIONS

- CPU Time = $\frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{avg. Cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$
- Time needed to complete N tasks using a k-stage Pipeline = $(k + N - 1)\,tp$, where tp = time per stage
- Speed Gain, $S = \frac{T'}{T} = \frac{Nk\,tp}{(k+N-1)\,tp}$. If $N \to \infty$ then, $S_\infty = \frac{k\,tp}{tp} = k$
- Overall Speed up because of Component = $\frac{\text{Execution time in old system}}{\text{Execution time in new system.}}$

$$= \frac{1}{1 - (\text{fraction of work done by component}) + \left(\frac{\text{fraction of work done by cmpt}}{\text{Component Speed up}}\right)}$$

- Disk Utilization = $\frac{\text{request arrival rate}}{\text{disk service rate}} = \frac{\text{requests per second}}{\text{I/O operations per second}}$
- Time Request spends in the queue = $\left[\frac{\text{disk utilisation}}{1 - \text{disk utilisation}}\right]$ (disk service time)
- Sequential EAT = HitRate * Access_c + (1 − HitRate) * (Access_c + Access_m)
- Parallel EAT = HitRate * Access_c + (1 − HitRate) * Access_m
- Speed Up = $\frac{\text{Memory Access time}}{\text{(Memory + Cache) Access EAT}}$, where EAT can be sequential or parallel.
- Speed Down = $\frac{EAT_{noVM}}{EAT_{with\ VM}}$

BINARY

* The complement systems only flip bits for negative numbers, positive numbers remain unchanged.

- Range of One's Complement → $[-(2^{N-1}-1), (2^{N-1}-1)]$
- Range of Two's Complement → $[-(2^{N-1}), (2^{N-1}-1)]$
- IEE-754 Single Precision | 1 | 8 | 23 | Bias of 127
- IEE-754 Double Precision | 1 | 11 | 52 | Bias of 1023 — Significand has an implied bit to the left.
- IEE 754 Single Precision: Min Positive Value = 1.0×2^{-127}, Max Positive Value = $(2 - 2^{-23}) \times 2^{127}$
- In Single Precision → Exponent of 255 → If significand = 0 → value = ± infinity; If significand ≠ 0 → value = NaN used to flag an err
- In Double Precision → Exponent of 2047
- Maximum error will be half of the smallest significand unit.
- In One's Compliment, if there is 1 in the carry while adding, then add it to the a

MEASURES

- Hertz → clock cycles per second (frequency)

$1\,MH_2 = 1{,}000{,}000\,H_2$. Processor speeds are measured in MH_2

- Byte → unit of Storage

$1\,kB = 2^{10}$ bytes, $1\,MB = 2^{20}$ bytes, $1\,GB = 2^{30}$ bytes.

Main Memory (RAM) is measured in MB or GB while disk storage is in GB or T

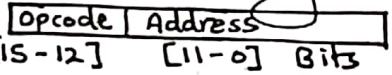
ASSEMBLY

The General Performance Equation:-

- $$\text{CPU Time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{avg Cycles}}{\text{Instruction}} \times \frac{\text{seconds}}{\text{Cycle}}$$

∴ CPU throughput can be improved by (i) Reducing no of instructions in a program, (ii) Reducing no of cycles per instruction and (iii) Reducing no of seconds per cycle.

- The following are characteristics of MARIE:-
  - → Binary two's compliment data representation
  - → Stored Program, fixed word length data and instructions
  - → 4k words of word-addressable main memory.
  - → 16 bit data words
  - → 16 bit instructions. 4 for the op-code, 12 for the address.
  - → 16-bit arithmetic logic unit (ALU)
  - → Seven Registers for control and data movement.
- The following are MARIE'S 7 REGISTERS:-
  - → Accumulator (AC) [16]: holds conditional or one operand of 2 operand instruction
  - → Memory Address Register (MBR) [16]: holds memory address after retrival from or before placement in memory.
  - → Memory Buffer Register (MAR) [12]: holds memory address of opeand of an instruction
  - → Program Counter (PC) [12]: holds the address of the next program instruction to be executed.
  - → Instruction Register (IR) [16]: holds an instruction immediately preceeding its execution
  - → Input Register (IN) [8]
  - → Output Register (OUT) [8]
- Format of MARIE Instruction



* When no operands are required address is 0.

- FUNDAMENTAL MARIE INSTRUCTIONS:-

Instruction No		Instruction	Meaning
Bin	Hex		
0001	1	Load X	Load contents of Address x into AC
0010	2	Store x	Store the contents of AC at address x
0011	3	Add x	Add contents of x to AC
0100	4	Subt x	Subtract contents of address x from AC
0101	5	Input	
0110	6	Output	
0111	7	Halt	Terminate program
1000	8	Skipcond	Skip next instruction on condition
1001	9	Jump x	Load the value of x into PC
1010	A	Clear	AC ← 0
1011	B	Addl x	Add value at address at address x to AC
0000	0	Jns x	Jumps and stores value of PC at address x then increments PC by 1
1100	C	Jumpl x	Jumps to address at x.

Load X
MAR ← X
MBR ← M[MAR]
AC ← MBR

- **Store X**
MAR ← X
MBR ← AC
M[MAR] ← MBR

- **Add X**
MAR ← X
MBR ← M[MAR]
AC ← AC+MBR

- **Subt X**
MAR ← X
MBR ← M[MAR]
AC ← AC-MBR

Jump X
PC ← X

- **Clear**
AC ← 0

- ~~Store x~~
- **Load I x**

MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC ← MBR

- **Add I X**

MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC ← AC + MBR

- **JnS X**

MAR ← X
MBR ← PC + 1
M[MAR] ← MBR
AC ← X + 1
PC ← X + 1

- **Jump I X**

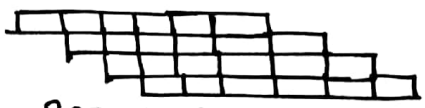
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
PC ← MBR

ISAs AND PIPELINES

Leftmost bit → MSB, Rightmost bit → LSB

Eg: 12345678

Address	00	01	10	11
Big Endian	12	34	56	78
Little Endian	78	56	34	12



PIPELINING

- Theoretical Speed-Up offerred by pipeline (T)

$$T = (k + N - 1)\, tp$$

[where k = number of stages, N = number of tasks, tp = time per stage.]

Time taken to complete N tasks using a k-stage pipeline.

- $\text{Speed-up } (S) = \dfrac{Nk\,tp}{(k+N-1)\,tp}$ If $n \to \infty$ then $S_\infty = \dfrac{k\,tp}{tp} = k$
- Pipeline Hazards: cause pipeline to stall or be flushed. Reasons:-
  (i) Resource conflicts (ii) Data dependancies (iii) Conditional branching
- Amdahl's Law: System performance depends not only on the performance of each of its components, but also on the fraction of work done by the component ϕ.

$$\text{System Speedup } (S_S) = \frac{1}{1 - \phi + (\phi / S_C)}$$

[S_C → component speed up]

BENCHMARKING

RISC Machines, CISC Machines
Reduced and Complex Instruction Set Computers.

→ RISC: → Few simple instructions fixed in length
- \+ Control units can be hardwired for maximum speed.
- \+ Instructions take same no of cycles to process → easier to manage and form pipelines.
- − More instructions required for complex operations.

→ Few addressing modes
- \+ Parameter passing through registers → fast sequential execution
- − Requires more CPU registers → expensive

- CISC → Many Complex instructions of variable length
  - − Control units must use microcode/special circuits to interpret instructions as they are fetched from memory → takes time.
  - − Instructions may take multiple cycles to process → harder to manage and form pipelines.
  - \+ Fewer instructions needed for complex operations.

→ many addressing modes
- − Parameter passing through memory → can be bottleneck
- \+ Fewer registers needed, fewer operands per instruction.

- Disk Utilisation: is the percentage of time that the disk will be busy servicing I/O requests.

$$t_q = \left(\frac{U_d}{1 - U_d}\right) t_{srv}$$

t_q → time a request spends in queue
t_{srv} → service time

$$U_d = \frac{r_{req}}{r_{srv}}$$

[r_{req} → Request arrival rate
r_{srv} → Disk service Rate]

MEMORY

THE MEMORY HIERARCHY

- Registers
- Cache memory
- Main memory
- Virtual memory (disk)

- Registers: Storage locations available on the processor.
- To access memory, CPU sends request to cache, if the memory is not in ca
  then the CPU sends it to main memory, then the request goes to disk.
- Once the data is located, the data and a number of its nearby data elemen
  are fetched into cache memory.
- Virtual memory is typically implemented using hard drive; it extends the
  address space from RAM to hard drive.
- Virtual Memory provides more space. Cache provides speed.
- Hit: when data is found - miss: when it is not found.
- Hit Rate: Percentage of time data is found - miss rate: percentage of time it is no
- Hit time: time required to access data - miss penalty: time required to process
- An entire block of data is copied after a hit → principle of locality:-
  (i) Temporal locality: Recently-accessed data elements tend to be accessed again
  (ii) Spatial locality: Accesses tend to cluster.
  (iii) Sequential Locality: Instructions tend to be accessed sequentially.
- tag → distinguishes cache memory block from another.
- valid bit → indicates whether cache block is being used.
- offset field → points to desired data in the block.

Total bits = Block bits + cache bits

Tag	Block	Offset

Size is determined by no of cache blocks
& determined by size of the block

- EAT (Effective Access Time) →

$$EAT_p = H * Access_c + (1-H) * Access_m$$
$$EAT_s = H * Access_c + (1-H) * (Access_c + Access_m)$$

* EAT takes all levels of memory into consideration

- Page Table Look Up Process
  → Extract page number from virtual address
  → Extract offset from virtual address
  → Go to page table and to get necessary frame number.
  → If frame number exists (page is in use), use corresponding frame number
  add the offset field to yield physical address.
  → If the page is not in main memory, generate a page fault and restar
  the access when the page fault is complete.
- Translation Look-aside buffer (TLB) → Special associative cache that stores th
  mapping of virtual pages to physical pages.
- TLB Lookup Process:-
  → Extract page number from virtual address
  → Extract the offset from virtual address
  → Search for virtual page number in TLB
  → If the (virtual page no, page frame no) pair is found in TLB, add the offset
  to the physical frame number and access memory location.
  → If there is a TLB miss, go to page table to get necessary frame num
  → If page is in memory, use the corresponding frame number and add of
  field to yield the physical address
  → If the page is not in main memory, generate a page fault and restart the ac
  when the page fault is complete.

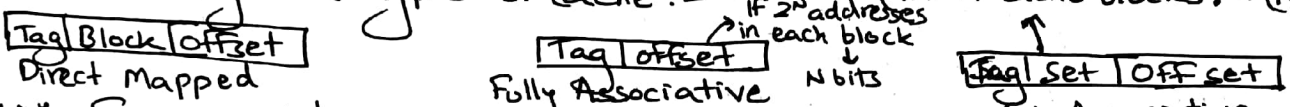
* When hard disk is accessed, data is updated → after refreshing there is
TLB hit and Cache hit.

- **Speed Up** → $\text{Speed Up} = \frac{\text{Memory Access Time}}{(\text{Memory+Cache}) \text{ Access Time (EAT)}}$

- $\text{Speed Down} = \frac{EAT_{noVM}}{EAT_{with\ VM}}$

* EAT wi
different f
and sequer

- Computer memory is organised in a hierarchy, with the smallest, fastest memory at the top and the largest, slowest memory at the bottom.
- Cache memory gives speed faster acesss to main memory, while virtual uses disk storage to give the illusion of having a large main memory
- Cache maps blocks of main memory to blocks of cache memory. Virtual maps page frames to virtual ~~memory~~ pages.
- There are 3 general types of cache:



- With fully associative cache and set associative cache, as well as virt memory replacement policies need to be established.
- Replacement policies include LRU, LIFO, LFU. These policies must also take account what to do with dirty blocks.
- All virtual memory must deal with fragmentation → internal for paged mer external for segmented memory.